Intel·ligència Artificial: Pràctica Planificació

Guillem Cabré, Carla Cordero, Hannah Röber

Curs 2024-25, Quadrimestre de tardor

Continguts

1 Introducció

El problema plantejat consisteix en la creació d'una eina capaç de planificar el visionat de continguts audiovisuals per a un usuari, tenint en compte diversos factors com el catàleg de continguts disponibles, les dependències entre aquests, els continguts ja visionats i aquells que es volen veure. L'objectiu és generar un pla que distribueixi els continguts al llarg dels dies.

La pràctica s'ha de desenvolupar amb el planificador *Fast Forward v2.3* i *Metric Fast Forward* i inclou diverses extensions que permeten iterar sobre diferents versions cada cop més complexes. Es posa especial èmfasi en la correcta modelització del domini i dels problemes, així com en la qualitat dels jocs de prova utilitzats. Finalment, la documentació i els resultats obtinguts seran avaluats en funció de la seva qualitat i adequació.

La pràctica es centra en resoldre un problema de planificació utilitzant el llenguatge de descripció PDDL. El nivell bàsic del problema implica la generació d'un pla de visionat on cada contingut pot tenir com a màxim un contingut predecessor, i no existeixen continguts paral·lels. El planificador ha de garantir que els continguts es visionen en l'ordre correcte, respectant aquestes dependències senzilles.

Les extensions incrementen la complexitat del problema afegint noves restriccions i funcionalitats:

- **Extensió 1:** Permet que un contingut tingui múltiples predecessors. El planificador ha de garantir que tots els predecessors es visionin abans del contingut corresponent.
- **Extensió 2:** Incorpora continguts paral·lels, que es poden visionar el mateix dia o en dies consecutius. Això reflecteix les relacions complexes entre subtrames o universos compartits.
- **Extensió 3:** Limita el nombre màxim de continguts a tres per dia.
- **Extensió 4:** Afegeix una durada en minuts als continguts i estableix un límit màxim de 200 minuts per dia, requerint un control més precís dels recursos temporals.

2 Modelatge del Domini

El domini **REDFLIX** s'ha creat per capturar les dependències entre continguts audiovisuals (pel·lícules, capítols de sèries) i les restriccions associades amb la seva assignació a dies en un pla de visionat. S'utilitzarà l'extensió de PDDL `:typing` per poder establir tipus de variables.

2.1 Variables

Les variables que representen el nostre domini són dues, aquestes seran subtipus de la variable `object`. Vegeu-les a continuació:

- `content`: Representa els continguts audiovisuals (capítols o pel·lícules).
- `day`: Representa els dies disponibles per al pla de visionat.

2.2 Predicats

Els predicats representen les relacions i restriccions entre els continguts i els dies i són la base per definir les precondicions i efectes de les accions. Mostrarem primer els predicats del problema amb **Nivell senzill**. I a continuació afegirem els predicats a mesura que avancem per les diferents extensions.

- `(watched ?c - content)`: Indica que un contingut `?c` ja s'ha vist i no cal assignar-lo al plan si es tracta d'algun predecessor o paral·lel d'algun contingut que es desitja veure. Aquest predicat és estrictament necessari per poder establir quins continguts del catàleg ja han sigut visionats i no afegir-los cíclicament en el pla de visionat.
- `(is_wanted ?c - content)`: Indica que l'usuari pot veure un contingut `?c`. Serà necessari per què serà la forma que tindrà l'usuari de marcar quin o quins continguts vol visionar, a partir d'aquest es navegarà fins afegir els altres continguts.
- `(predecessor ?c1 - content ?c2 - content)`: Indica que el contingut `?c1` és predecessor de `?c2`. Per tant, `?c1` s'ha de veure abans que `?c2`. Aquest predicat serà l'encarregat de permetre la navegació entre continguts predecessors.
- `(day_to_watch ?c - content ?d - day)`: Assigna un contingut `?c` a un dia `?d`. Estableix quin dia s'ha de visionar el contingut en el pla de visionat.
- `(yesterday ?d1 - day ?d2 - day)`: Indica que `?d1` és el dia anterior a `?d2`. Estableix la relació entre dos dies, d'aquesta manera podrem respectar que un contingut predecessor s'hagi de visualitzar el dia abans que el successor.
- `(assigned ?c - content)`: Marca que un contingut `?c` ja té un dia assignat per evitar que s'assigni a més de un dia. Aquest predicat no és estrictament necessari, però ens redueix la complexitat del sistema, podriem utilitzar un `exists` de `day_to_watch` del contingut `?c`. Aquesta operació tindria un cost molt elevat en precondicions, per això hem optat per aquesta predicat auxiliar.

Aquests són els predicats del **Nivell bàsic**, ara enumerarem els predicats de les altres extensions:

Extensió 1: Els predicats romanen iguals, l'únic que canviarà serà la post condició quan s'assigni un contingut a un dia.

Extensió 2: S'afegeix el predicat de contingut paral·lel (`parallel`).

- `(parallel ?c1 - content ?c2 - content)`: Estableix que `?c1` és paral·lel amb `?c2` i s'han de veure el mateix dia o en dies consecutius. Hem assumit la relació paral·lel és unidireccional, però el nostre model permet que sigui bidireccional, si en el problema s'escriuen:

```
(parallel ?c1 ?c2)
(parallel ?c2 ?c1)
```

Extensió 3: Apareixen tres predicats nous, cada un indicant quin dels tres continguts de cada dia han sigut ja assignats. En comptes d'afegir tres predicats nous podríem haver optat per usar el paquet de PDDL `:fluents` per comptabilitzar el número de continguts que s'han assignat en un dia. Però vam optar per aquesta solució perquè el nombre d'assignacions per dia estava fitat a tres que és un número petit. A més, d'aquesta manera, fèiem que el programa continués funcionant amb `Fast Forward v2.3.` i no haver d'optar per un altre planificador automàtic. Els tres predicats nous són:

- `(assigned_one ?d - day)`: S'ha assignat el primer contingut el dia `?d`.
- `(assigned_two ?d - day)`: S'ha assignat el segon contingut el dia `?d`.
- `(assigned_three ?d - day)`: S'ha assignat el tercer contingut el dia `?d`.

Extensió 4: Els tres predicats de l'extensió anterior són retirats. Tampoc s'afegeix cap predicat nou en aquesta extensió. En aquesta extensió es farà ús de fluents, s'explicarà a continuació.

2.3 Funcions

Les funcions introdueixen fluents per gestionar informació numèrica. Aquestes només es fan servir per l'Extensió 4:

- `(priority ?d - day)`: Indica la prioritat assignada a un dia, per poder optimitzar la distribució dels continguts en dies.
- `(total-days)`: Enregistra la suma de les prioritats dels dies utilitzats. Necessari per optimitzar el pla en la mètrica i que s'assignin els continguts prioritzant els primers dies i consecutius perquè no hi hagi dies entremig sense contingut o que s'assignin als últims dies dels disponibles.
- `(duration ?c - content)`: Assigna la durada de cada contingut, modelant restriccions de temps.
- `(day_duration ?d - day)`: Conté la durada acumulada dels continguts assignats a un dia.
- `(remaining-content)`: Indica el nombre de continguts desitjats encara no assignats, permetent controlar l'avanç cap a l'objectiu.

2.4 Accions

En totes les extensions hem fet ús de tres accions. Les tres accions descrites, add_content, set_day_unique i set_day, són fonamentals per garantir el funcionament del programa. Primer de tot per afegir els continguts desitjats al pla i, per altra banda, per gestionar l'assignació de dies als continguts, diferenciant entre aquells que són independents i els que tenen relacions amb altres continguts.

Es va dividir l'acció d'assignació de dies en dos operadors (set_day_unique i set_day) per gestionar de manera específica continguts amb dependències i sense, simplificant així les precondicions i efectes de cada acció.

Per explicar cada una d'aquestes accions farem una breu desprescripció del seu funcionament i explicarem les diferents iteracions que han tingut al llarg de les extensions. Per simplificar, direm que tenim dos planificadors diferents, el de la **Extensió 3** i el de la **Extensió 4**, si les accions d'aquests dos planificadors són diferents, analitzarem en què canvien.

Les tres accions són les següents:

add_content

Aquesta acció afegeix al pla de visionat continguts que l'usuari vol veure, o que el sistema vol que l'usuari vegi.

- **Justificació de l'operador**: Sense aquesta acció, el sistema no podria garantir que tots els continguts relacionats s'afegeixin de manera automàtica al pla un cop els seus predecessors ja han estat vistos.
- **Precondicions**: El contingut no s'ha vist i és desitjat (`is_wanted`).

- **Efectes**: Inclou els predecessors o continguts paral·lels a la llista de continguts desitjats.

```
;; Anade un contenido al plan sin visionado
(:action add_content
  :parameters (?c - content)
  :precondition (and
    (is_wanted ?c)
    (not (watched ?c)))
  :effect (and
  ;; Si el contingut vist te un successor o algun contingut en parallel per veure i no ha estat vist, s'afegeix a la llista de continguts per veure
    (forall (?c2 - content)
      (when (and
        (or (predecessor ?c2 ?c)(parallel ?c2 ?c))
        (not (watched ?c2))
        (not (is_wanted ?c2)))
          (is_wanted ?c2)
      )
))
```

Listing 1: Acció add_content

set_day_unique

Assigna un dia a un contingut que no té successors ni és paral·lel amb altres.

- **Justificació de l'operador**: Aquesta acció permet assignar un dia a continguts no tenen successors ni paral·lels. És crucial per garantir que els continguts sense relacions siguin assignats de manera eficient.
- **Precondicions**: El contingut és desitjat, no s'ha vist, no està assignat, el dia té espai disponible (no més de tres continguts), i no té successors ni paral·lels.
- **Efectes**: Marca el contingut com a assignat i reserva el dia corresponent.

```
;; Establece un dia a un contenido del plan de visionado si este no es predecesor ni paralelo de otros.
(:action set_day_unique
  :parameters (?c - content ?d - day)
  :precondition (and
    (is_wanted ?c)
    (not (watched ?c))
    (not (assigned ?c))
    (or
      (not (assigned_one ?d))
      (not (assigned_two ?d))
      (not (assigned_three ?d)))
    ;; Verifica que no haya sucesores ni contenidos paralelos que no hayan sido vistos
    (not (exists (?c2 - content)
      (and
        (or (predecessor ?c ?c2) (parallel ?c ?c2))
        (not (watched ?c2))
        (is_wanted ?c2)))
    )
  :effect (and
    (day_to_watch ?c ?d)
    (assigned ?c)
    (when (not (assigned_one ?d)) (assigned_one ?d))
    (when (and (assigned_one ?d) (not (assigned_two ?d))) (assigned_two ?d))
    (when (and (assigned_one ?d) (assigned_two ?d) (not (assigned_three ?d))) (assigned_three ?d))
```

```
25  ))
```

Listing 2: Acció set_day_unique

Per l'**Extensió 2** s'afegeixen les línies que verifiquen que no hi hagi continguts paral·lels que no hagin sigut vistos (línia 15). Després, per l'**Extensió 3**, s'afegeixen les línies de la precondició `or` de `(not (assigned_<number> ?d))`, i les de la postcondició també (línies 8-11 i 22-24).

Per altra banda, l'**Extensió 4** elimina els canvis fets per la tercera extensió i afegeix a la precondició:

`(<= (+ (day_duration ?d) (duration ?c)) 200)`: s'assegura que el dia `?d` no se li assignin més de 200 minuts.

També es modifica la postcondició, a la qual se li afegeixen les tres següents línies:

`(increase (day_duration ?d) (duration ?c))`: incrementa en duració de continguts visionats en el dia `?d` en duració de `?c` minuts de visionat.

`(increase (total-days) (priority ?d))`: incrementa la prioritat del dia en que s'ha assignat el contingut.

`(decrease (remaining-content)` 1): decrementa els continguts per veure

set_day

Assigna un dia a un contingut que pot tenir predecessors o paral·lels.

- **Justificació de l'operador**: Aquesta acció permet gestionar les relacions temporals i garantir que els continguts es visualitzin en l'ordre correcte. Sense aquesta acció, el planificador no podria garantir que els continguts amb dependències es distribueixin correctament al llarg del pla, respectant l'ordre i les restriccions definides.
- **Precondicions**: El contingut és desitjat, no està assignat, compleix amb les relacions de predecessors o paral·lels, i el dia té espai disponible.
- **Efectes**: Assigna el contingut al dia seleccionat, respectant les restriccions.

```
1   ;; Establece un dia a un contenido del plan de visionado si este es
       predecesor o paralelo. El contenido al que se le asignara un dia es el
       ?c1 y el dia del asignado es ?d1. ?c2 y ?d2 son el contenido y dia del
       sucesor o paralelo del contenido que vamos a asignarle un dia.
2   (:action set_day
3     :parameters (?c1 ?c2 - content ?d1 ?d2 - day)
4     :precondition (and
5       (is_wanted ?c1)
6       (not (assigned ?c1))
7       (day_to_watch ?c2 ?d2)
8       (or
9         (not (assigned_one ?d1))
10        (not (assigned_two ?d1))
11        (not (assigned_three ?d1)))
12      (or
13        ;condiciones para contenido paralelo
14        (and
15          (parallel ?c1 ?c2)
16          (or
17            (previous ?d1 ?d2)
18            (= ?d1 ?d2)))
19        ;condiciones para contenido predecessor
20        (and
21          (predecessor ?c1 ?c2)
22          (previous ?d1 ?d2))))
23    :effect (and
```

```
24        (day_to_watch ?c1 ?d1)
25        (assigned ?c1)
26        (when (not (assigned_one ?d1)) (assigned_one ?d1))
27        (when (and (assigned_one ?d1) (not (assigned_two ?d1))) (assigned_two
              ?d1))
28        (when (and (assigned_one ?d1) (assigned_two ?d1) (not (assigned_three
              ?d1))) (assigned_three ?d1))
29    ))
```

Listing 3: Acció set_day

Per l'**Extensió 2** s'afegeixen les línies amb la condició dels continguts paral·lels (línies 15-18). Després per l'**Extensió 3**, s'afegeixen les línies de la precondició `or` de `(not (assigned_<number> ?d))`, i les de la postcondició també (línies 8-11 i 26-28).

Els canvis de l'**Extensió 4** son idèntics als de l'acció anterior: eliminar els canvis de l'extensió 3 i afegir les línies a la precondició i postcondició:

`(<= (+ (day_duration ?d) (duration ?c)) 200)`: s'assegura que el dia `?d` no se li assignin més de 200 minuts.

`(increase (day_duration ?d) (duration ?c))`: incrementa en duració de continguts visionats en el dia `?d` en duració de `?c` minuts de visionat.

`(increase (total-days) (priority ?d))`: incrementa la prioritat del dia en que s'ha assignat el contingut.

`(decrease (remaining-content)` 1): decrementa els continguts per veure

3 Modelatge dels Problemes

Els problemes específics es modelen com a instàncies del domini **Redflix**, definint els objectes, l'estat inicial i l'estat final.

3.1 Objectes

Els objectes representen els elements que intervenen en el problema, en aquest cas es defineixes dos tipus principals: continguts (representen els episodis, pel·lícules o capítols de la plataforma) i dies (que representen el temps per visionar els continguts).

Per exemple, si es vol modelar la visualització de cinc continguts durant cinc dies, es definirien així:

```
(:objects
  lalaland pulpfiction theprestige dark - content ;; Continguts disponibles
  day1 day2 day3 day4 day5 - day ;; Dies assignables
)
```

Listing 4: Definició d'objectes

Aquesta definició proporciona la base per establir les relacions i restriccions del problema, com ara les dependències entre continguts o la disponibilitat de dies.

3.2 Estat Inicial

L'estat inicial descriu les relacions entre continguts i dies, incloent-hi:

- `Continguts ja vistos:` episodis que l'usuari ja ha visualitzat i no cal incloure en el pla. Representat amb el predicat `(watched ?c - content)`.
- `Continguts que l'usuari vol veure:` Llista dels continguts que l'usuari pot veure. Representat amb el predicat `(is_wanted ?c - content)`.
- `Relacions de predecessors i paral·lels:` Predecessors que han de ser vistos abans d'un altre contingut i continguts paral·lels que poden ser vistos el mateix dia. Representat amb el predicat `(predecessor ?c1 - content ?c2 - content)`.
- `Relacions temporals entre els dies:` Ordre cronològic dels dies disponibles. Representat amb el predicat `(yesterday ?d1 - day ?d2 - day)`.

Un exemple d'inicialització és el següent, on bb_s_i es la i-èssima temporada de *Breaking Bad* (per posar un exemple) i cc_i és el i-èssim spin-off de la sèrie (per exemple: *El Camino* i *Better Call Saul*). El spin-off estarà relacionat amb la temporada de la sèrie amb la que s'assigni que és paral·lel:

```
(:init
  (is_wanted bb_s3) ;; L'usuari vol visiionar bb_s3
  (watched bb_s1) ;; L'usuari ha vist bb_s1
  (watched bb_s2) ;; L'usuari ha vist bb_s2
  (predecessor bb_s1 bb_s2) ;; bb_s1 es predecesor de bb_s2
  (predecessor bb_s2 bb_s3) ;; bb_s2 es predecesor de bb_s3
  (parallel cc1 cc2) ;; cc1 es un contingut parallel de cc2
  (yesterday day1 day2) ;; day1 es anterior a day2
  (yesterday day2 day3) ;; day2 es anterior a day3
)
```

Listing 5: Exemple d'inicialització del problema

3.3 Estat Final

L'estat objectiu s'ha d'assegurar que els continguts estan assignats a dies concrets dins de les limitacions del problema. Un altre exemple d'estat final és marcar que aquells continguts han estat assignats, sense marcar al planificador quin dia s'haurà de veure d'aquesta manera el propi planificador decidirà un dia pel quan es compleixin totes els restriccions.

Exemple d'estat final:

```
(:goal (and
  (day_to_watch bb_s3 day3) ;; bb_s3 es vol que es visualitzi el dia 3
  (day_to_watch cc1 day2) ;; cc1 el dia 2
  (day_to_watch cc2 day2) ;; c2 tambe el dia 2
))
```

Listing 6: Exemple de *goal* del problema

3.4 Mètrica

En l'extensió 4 al utilitzar *Metric Fast Forward* hi ha l'opció d'afegir una mètrica que ens serveix per a que busqui un pla que optimitzi un paràmetre del problema. En el nostre cas volem que assigni el màxim de continguts per dia i que s'assignin als primers dies i consecutius, per lo que s'han assignat prioritats als dies i `total-days` conté la suma de les prioritats dels dies en que s'assignen els continguts per lo que es vol minimitzar aquest paràmetre. Per lo que afegim la següent línia de codi al problema:

```
(:metric minimize (total-days))
```

Listing 7: Exemple de *mètrica* del problema

4 Desenvolupament del Model

El model s'ha desenvolupat iterativament, abordant les extensions de manera progressiva. Els canvis de cada versió ja s'han anat explicant en els apartats anteriors en més detalls per cada subsecció. Aquí enumerarem i donarem més detalls sobre els cavis que no s'hagin acabat d'explicar en els apartats anteriors.

Nivell Bàsic

Es va començar modelant continguts que tinguessin com a màxim un predecessor. Les accions no consideraven continguts paral·lels ni límits diaris.

Extensió 1

Tal i com vam plantejar el **Nivell Bàsic**, no vam fer canvis en aquesta extensió. L'únic canvi, per així dir-ho va ser en el fitxer del problema. El nivell bàsic ja tenia el fragment de codi del següent listing (8). Per tant el problema del Nivell Bàsic tenia cadenes de predecessors que formaven un camí (entenent-ho com en teoria de grafs, on els nodes son continguts i les arestes marquen els predecesors), i la extensió 1 ho gestiona com a arbres on cada node pot tenir més d'un predecessor.

```
(forall (?c2 - content)
  (when ((predecessor ?c2 ?c))
    (is_wanted ?c2)
  )
)
```

Listing 8: Fragment per assignar més d'un predecessor com a pendent de veure

Extensió 2

A l'extensió 2, es va introduir la capacitat de modelar continguts paral·lels, permetent que alguns continguts es puguin veure el mateix dia o en dies consecutius. Aquesta característica va requerir l'addició del predicat `parallel` per definir la relació entre continguts paral·lels i noves restriccions a les accions d'assignació de dies (`set_day` i `set_day_unique`).

Es va afegir el predicat `parallel`, que indica si dos continguts es consideren paral·lels:

```
(:predicates
  ;; ...
  (parallel ?c1 - content ?c2 - content)
)
```

Listing 9: Definició del predicat `parallel`

Les accions d'assignació de dies es van modificar per assegurar que, si dos continguts són paral·lels, aquests es programin el mateix dia o en dies consecutius. Aquesta restricció es va afegir a les precondicions de les accions, com es mostra a continuació:

```
;; PRECONDICIO
;; ...
(or
  (and
    (parallel ?c1 ?c2)
    (day_to_watch ?c2 ?d2)
    (or
      (yesterday ?d ?d2)
      (= ?d ?d2)))
  ;; condicions per continguts predecessors
)
```

Listing 10: Restriccions per continguts paral·lels

Per acabar, al fitxer problema, es va afegir la informació sobre les relacions de paral·lelisme entre continguts, així com la relació temporal entre dies. Per exemple:

```
(:init
  ;; ...
  (cc1 cc2)
  ;; ...
  (yesterday day1 day2)
  (yesterday day2 day3)
)
```

Listing 11: Inicialització de paral·lels i dies consecutius

Extensió 3

Es van afegir restriccions per limitar el nombre de continguts per dia a tres. Això es va aconseguir amb els predicats `assigned_one`, `assigned_two` i `assigned_three`.

```
(:predicates
  ;; ...
  (assigned_one ?d - day)
  (assigned_two ?d - day)
  (assigned_three ?d - day)
)
```

Listing 12: Definició del predicat `assigned_<one,two,three>`

En les precondicions d'assignar un dia, tant en `set_day` com a `set_day_unique` s'afegeixen les següents linies:

```
;; PRECONDICIO
;; ...
(or
  (not (assigned_one ?d))
  (not (assigned_two ?d))
  (not (assigned_three ?d))
)
;; POSTCONDICIO
;; ...
(when (not (assigned_one ?d)) (assigned_one ?d))
(when (and (assigned_one ?d) (not (assigned_two ?d))) (assigned_two ?d))
(when (and (assigned_one ?d) (assigned_two ?d) (not (assigned_three ?d)))
    (assigned_three ?d))
```

Listing 13: Fragment per limitar a 3 els continguts per dia

A la precondició es valida que pel dia `?d` hi hagi algun *slot* disponible. I a la postcondició, s'assigna al primer *slot* disponible el contingut `?c`.

El fitxer problema no es va veure modificat perquè per defecte, els nous predicats han d'estar marcats com a no-assignats. Ja que el programa PDDL serà el que vagi assignant-los un per un.

Extensió 4

A l'extensió 4, es va haver d'incorporar una nova característica al domini per gestionar la duració dels continguts i assegurar que no es superessin els 200 minuts de visualització per dia. Aquesta funcionalitat es va implementar utilitzant una funció `day_duration` que acumula els minuts assignats a cada dia, així com la duració específica de cada contingut mitjançant la funció `duration`.

Per a controlar la limitació de minuts per dia, es van modificar les precondicions i els efectes de les accions d'assignació de dies (`set_day` i `set_day_unique`). Es va assegurar que només es poguessin assignar continguts a un dia si la seva duració, sumada a la duració acumulada d'aquell dia, no superava els 200 minuts.

```
;; PRECONDICIO
(<= (+ (day_duration ?d) (duration ?c)) 200)
```

```
3
4   ;; POSTCONDICIO
5   (increase (day_duration ?d) (duration ?c))
```

Listing 14: Restriccions per controlar el límit de 200 minuts per dia

Aquestes línies garanteixen que:

- A la precondició, es comprova que el contingut `?c` es pugui afegir al dia `?d` sense superar el límit de 200 minuts.
- A la postcondició, un cop assignat, la funció `day_duration` s'actualitza sumant la duració del contingut `?c` al total acumulat.

També es va definir la funció `duration` per a cada contingut i es va afegir la funció `day_duration` per emmagatzemar la suma total de minuts assignats a cada dia:

```
1   (:functions
2     (duration ?c - content)
3     (day_duration ?d - day)
4   )
```

Listing 15: Funcions per l'extensió 4

Es va afegir al fitxer problema la duració de cada contingut, així com la inicialització dels valors de `day_duration` a 0, que guardaran posteriorment el total acumulat i de `priority` a la corresponent segons el dia (més prioritat als primers dies i valors petits). Per exemple:

```
1   (:init
2     ;; ...
3     (= (duration bb_s1) 45)
4     (= (duration bb_s2) 50)
5     (= (duration bb_s3) 60)
6     (= (duration cc1) 30)
7     (= (duration cc2) 25)
8
9     (= (day_duration day1) 0)
10    (= (day_duration day2) 0)
11    (= (day_duration day3) 0)
12
13    (= (priority day1) 1)
14    (= (priority day2) 2)
15    (= (priority day3) 3)
16  )
```

Listing 16: Novetats en la inicialització per l'extensió 4

5 Conclusió

El desenvolupament d'aquest projecte ens ha proporcionat una oportunitat per posar en pràctica els coneixements adquirits sobre planificació automàtica i modelatge mitjançant PDDL. Hem pogut aplicar de manera progressiva els conceptes teòrics apresos per finalment desenvolupar un sistema funcional que compleix els requisits establerts.

El treball ha estat un repte, sobretot a l'hora d'implementar les diferents extensions i garantir que totes les funcionalitats fossin compatibles. A través d'aquest procés, hem millorat la nostra comprensió dels sistemes de planificació.

5.1 Coneixements adquirits

Aquest projecte ens ha permès aprofundir en diversos aspectes relacionats amb la planificació, com ara:

- La definició de dominis i problemes mitjançant PDDL, amb una comprensió clara de com estructurar accions, precondicions i efectes.
- L'ús de predicats, funcions i restriccions per modelar situacions reals de manera precisa i eficient.
- La importància de modularitzar el desenvolupament per incorporar funcionalitats progressivament sense perdre consistència.
- L'ús de planificadors automàtics per resoldre problemes amb limitacions complexes i obtenir solucions viables.

També hem après a analitzar els resultats obtinguts (i errors que hem tingut) i a identificar possibles limitacions, fet que ens ha ajudat a entendre millor les capacitats i les limitacions dels sistemes de planificació.

5.2 Possibles millores

Tot i que hem assolit els objectius establerts, creiem que aquest projecte es podria millorar en diversos aspectes:

- **Major flexibilitat**: Introduir més opcions per als usuaris, com ara prioritzar continguts o ajustar limitacions personalitzades.
- **Eines de depuració**: Incorporar mecanismes per visualitzar més fàcilment les relacions entre continguts i els plans generats.
- **Anàlisi posteriors**: Afegir funcionalitats que permetin analitzar plans generats i obtenir estadístiques sobre la seva eficiència o sobre el compliment de les preferències.
- **Adaptació a altres contexts**: Explorar com aquest model es podria adaptar a altres escenaris, com la planificació de tasques personals o professionals.
- **Problema més ampli**: no només limitar-nos a cadenes de continguts paral·lels i predecessors, tenir formes d'afegir en el pla de visionat també continguts que es relacionin amb els gustos del client.

En resum, aquest projecte ha estat una experiència enriquidora que ens ha ajudat a consolidar els nostres coneixements i a desenvolupar habilitats pràctiques en l'àmbit de la planificació automàtica.

A Annexos - Especificació PDDL

A.1 Especificació del domini en PDDL

A continuació, es mostra l'especificació completa del domini utilitzat en aquest projecte:

```
(define (domain example-domain)
  (:predicates
    (on ?x ?y) ;; Descripcio del predicat
    ;; ...
  )
  (:action move
    :parameters (?x ?y)
    :precondition (and (clear ?y) (on ?x table))
    :effect (and (not (on ?x table)) (on ?x ?y))
  )
)
```

Listing 17: Especificació del domini en PDDL

A.2 Especificació del problema en PDDL

Aquí s'inclou la descripció del problema utilitzat per validar el domini:

```
(define (problem example-problem)
  (:domain example-domain)
  (:objects
    block1 block2 block3 table
  )
  (:init
    (on block1 table)
    (on block2 table)
    ;; ...
  )
  (:goal
    (and (on block1 block2) (on block2 block3))
  )
)
```

Listing 18: Especificació del problema en PDDL

B Annexos - Intèrpret FF

B.1 Intèrpret per a sortides de FF i MFF

Aquest programa en Python està dissenyat per interactuar amb un planificador automatitzat, específicament l'executable FF (*Fast Forward* o *Metric Fast Forward*), amb l'objectiu de generar i analitzar plans per a la plataforma de continguts "Redflix". Mitjançant una interfície de consola, l'usuari pot seleccionar entre diferents extensions del domini PDDL (Planning Domain Definition Language), cadascuna amb característiques addicionals, com ara restriccions de predecessors, accions paral·leles o límits de durada. El programa també permet seleccionar fitxers de problema de manera interactiva mitjançant un quadre de diàleg i executar el planificador FF sobre els fitxers seleccionats. Per als problemes més avançats, a l'extensió 4, s'inclou un generador opcional que crea instàncies de problemes aleatoris amb configuracions específiques.

Un cop executat el planificador, el programa filtra la sortida de FF per identificar accions rellevants del pla generat, com ara afegir contingut ("ADD_CONTENT") i establir dies únics ("SET_DAY_UNIQUE") o compartits ("SET_DAY"). Aquestes accions es processen i s'organitzen en un diccionari, on les claus representen dies i els valors són llistes de continguts planificats per a cada dia. Finalment, el programa mostra a la consola un resum clar i estructurat del pla, permetent a l'usuari analitzar la planificació proposada i les decisions de l'algorisme de manera eficient.

Algorisme 1 Intèrpret de sortides de FF i MFF

```
1: Entrada: Sortida del planificador (stdout)
2: Sortida: Pla organitzat per dies
3: Inicialitzar plan_by_day com un diccionari buit
4: Definir patrons per a les accions ADD_CONTENT, SET_DAY_UNIQUE, i SET_DAY
5: for cada línia en la sortida del planificador do
6:     Eliminar espais innecessaris i prefixos de la línia
7:     if la línia conté ADD_CONTENT then
8:         Extraure el contingut i guardar-lo
9:     else if la línia conté SET_DAY_UNIQUE then
10:        Extraure contingut i dia, i afegir-ho a plan_by_day
11:    else if la línia conté SET_DAY then
12:        Extraure contingut i dia, i afegir-ho a plan_by_day
13:    end if
14: end for
15: Retornar plan_by_day
```

C Annexos - Jocs de Prova

En aquest apartat, presentem els jocs de prova dissenyats per a avaluar el model del problema desenvolupat en PDDL per a les diverses extensions. Aquests jocs de prova consisteixen en conjunts de problemes escollits que simulen diferents escenaris del domini, variant en complexitat i restriccions per provar diversos casos. Alguns d'aquests jocs de prova han estat generats per un fitxer en Python que genera problemes de manera aleatòria, el qual s'explica a continuació. L'objectiu d'aquests casos de prova és avaluar el bon comportament del model enfront de diverses configuracions i relacions entre els elements del problema. Els problemes generats permeten validar tant la correcció del model com la seva adequació als requisits plantejats.

C.1 Generador de problemes

El programa en Python genera automàticament 5 fitxers de problemes en format PDDL de manera aleatòria i els guarda en una carpeta que crea en el mateix directori on es troba aquest fitxer. Els problemes generats són útils per a provar i avaluar algorismes de planificació automàtica, ja que ofereixen escenaris variats i bastants exemples per provar el model en una àmplia varietat de configuracions; a més a més, estan creats per a que es pugui obtenir un pla en la majoria dels casos generats. Detalls del programa:

Estructura del problema:
Cada problema inclou un conjunt de continguts (contents) que en genera un nombre aleatori i un conjunt de dies (days) de quantitat aleatòria al voltant del nombre de continguts que s'han de planificar per assegurar que hi hagi un plan. Els continguts es classifiquen uns quants en `is_wanted` del conjunt de continguts d'abans de manera aleatòria i es generen un nombre aleatori petit de nous continguts que seran `watched`.

Generació de dades i relacions:
Les relacions entre els continguts es generen aleatòriament, assegurant que no hi hagi redundàncies ni conflictes entre les relacions de precedència i paral·lelisme. A cada contingut se li assigna una durada aleatòria, representant el temps necessari per a consumir-lo.

Inicialització de l'estat:
En la secció :init del problema, s'especifiquen totes les relacions de precedència i paral·lelisme, els continguts desitjats, els continguts ja vists, les durades dels continguts, i les relacions temporals entre els dies (per exemple, quin dia precedeix a un altre). També s'inicialitzen mètriques com day_duration i remaining-content que es calcula.

Definició de l'objectiu i mètrica:
Es posa també l'objectiu i la mètrica a optimitzar que és igual sempre.

C.2 Nivell Bàsic

Joc de prova 1

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint continguts predecessors en cadena, afegint-los al pla també complint amb les restriccions de precedència

Entrada: Tenim 10 continguts en total, es desitgen veure 2 continguts. Hi ha 5 dies disponibles. Un contingut té 4 continguts predecessors en cadena i l'altre també té un altre predecessor. En el *listing* d'avall es pot veure el codi del problema. detalladament(19)

Sortida esperada: S'espera que el planificador assigni per veure els 5 continguts en precedència encadenada als 5 dies consecutius i els altres dos continguts en dos dies consecutius qualssevol.

Sortida obtinguda: Les restriccions de precedència es compleixen. Estan afegits al pla tots els continguts desitjats amb els seus predecessors. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: BB_S1, CC1
DAY2: CC2, BB_S2
DAY3: BB_S3
DAY4: BB_S4
DAY5: BB_S5
```

Figura 1: Sortida de FFInterpreter.py (B.1) del NB-J1.

Joc de prova 2

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint com a molt un predecessor als continguts desitjats per veure, afegint-los al pla també i que es respectin les precedències de visualització. Veure que es poden assignar continguts sense límit a un dia.

Entrada: Tenim 9 continguts totals, dels quals 6 es volen veure, 2 ja s'han vist, un dels dos és predecessor d'un contingut desitjat i un altre a part predecessor d'un altre desitjat. Només hi ha 2 dies disponibles. En el *listing* d'avall es pot veure el codi del problema detalladament(20)

Sortida esperada: El planificador hauria d'assignar en els 2 dies els dos continguts predecessors i la resta que no s'han vist i es volen repartits entre els dos dies ja que no hi ha límit per dia.

Sortida obtinguda: Les restriccions de precedència es compleix, no s'assignen els que ja s'han vist i s'han assignat al primer dia la resta de continguts. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: A3, A5, A6, A7, A8, B1
DAY2: A4
```

Figura 2: Sortida de FFInterpreter.py (B.1) del NB-J2.

```
(define (problem
    redflix-problem)
(:domain redflix)

(:objects
bb_s1 bb_s2 bb_s3 bb_s4 bb_s5
    aa1 aa2 aa3 cc1 cc2 -
    content
day1 day2 day3 day4 day5 -
    day)

(:init
(is_wanted bb_s5)
(is_wanted cc2)

(predecessor bb_s1 bb_s2)
(predecessor bb_s2 bb_s3)
(predecessor bb_s3 bb_s4)
(predecessor bb_s4 bb_s5)
(predecessor cc1 cc2)

;; Dias anteriores
(yesterday day1 day2)
(yesterday day2 day3)
(yesterday day3 day4)
(yesterday day4 day5))

(:goal (and
(day_to_watch bb_s1 day1)
(assigned cc1)))
)
```

Listing 19: Joc de Prova 1 - Nivell Bàsic

```
(define (problem
    redflix-problem)
(:domain redflix)

(:objects
a1 a2 a3 a4 a5 a6 a7 a8 b1 -
    content
day1 day2 - day)

(:init
(watched a1)
(watched a2)

(is_wanted a3)
(is_wanted a4)
(is_wanted a5)
(is_wanted a6)
(is_wanted a7)
(is_wanted a8)

(predecessor b1 a4)
(predecessor a1 a3)

(yesterday day1 day2))

(:goal (and
(assigned a3)
(assigned a4)
(assigned a5)
(assigned a6)
(assigned a7)
(assigned a8)
(assigned b1)))
)
```

Listing 20: Joc de Prova 2 - Nivell Bàsic

C.3 Extensió 1

Joc de prova 1

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint diversos continguts predecessors els continguts desitjats, afegint-los al pla també complint amb les restriccions de precedència. Veure que es poden assignar continguts sense límit a un dia.

Entrada: Tenim 10 continguts en total, es desitgen veure 2 continguts. Hi ha només 2 dies disponibles. Un contingut té 4 continguts predecessors en cadena i l'altre també té 2 altres predecessor. En el *listing* d'avall es pot veure el codi del problema. detalladament(21)

Sortida esperada: S'espera que el planificador assigni tots els continguts predecessors, que son 6, dels continguts desitjats al dia 1 i els desitjats al dia 2.

Sortida obtinguda: Les restriccions de precedència es compleixen. Estan afegits al pla tots els continguts desitjats amb els seus predecessors. Veiem que al dia 1 s'afegeixen tots els predecessors sense límit. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: CC1, AA1, BB_S1, BB_S2, BB_S3, BB_S4
DAY2: BB_S5, CC2
```

Figura 3: Sortida de FFInterpreter.py (B.1) del E1-J1.

Joc de prova 2

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint diversos continguts predecessors els continguts desitjats, afegint-los al pla també complint amb les restriccions de precedència, sense que s'afegeixin els que ja s'han vist.

Entrada: Tenim 8 continguts totals, dels quals 2 es volen veure, 3 ja s'han vist, dos dels tres són predecessor d'un contingut desitjat. Hi ha 5 dies disponibles. En el *listing* d'avall es pot veure el codi del problema detalladament(22)

Sortida esperada: El planificador hauria d'assignar en un dia un contingut desitjat i el dia anterior els seus predecessors que no s'han vist i el contingut desitjat que queda en qualsevol dia.

Sortida obtinguda: Les restriccions de precedència es compleix, no s'assignen els que ja s'han vist i s'han assignat al primer dia els que son predecessors i entre els dos primers dies els continguts desitjats. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: CC2, BB_S1, BB_S2
DAY2: BB_S5
```

Figura 4: Sortida de FFInterpreter.py (B.1) del E1-J2.

```
(define (problem
    redflix-problem)
(:domain redflix)

(:objects
bb_s1 bb_s2 bb_s3 bb_s4 bb_s5
    aa1 aa2 aa3 cc1 cc2 -
    content
day1 day2 - day)

(:init
(is_wanted bb_s5)
(is_wanted cc2)

(predecessor bb_s1 bb_s5)
(predecessor bb_s2 bb_s5)
(predecessor bb_s3 bb_s5)
(predecessor bb_s4 bb_s5)
(predecessor cc1 cc2)
(predecessor aa1 cc2)

;; Dias anteriores
(yesterday day1 day2))

(:goal (and
(assigned bb_s1)
(assigned bb_s2)
(assigned bb_s3)
(assigned bb_s4)
(assigned cc1)
(assigned aa1)))
)
```

Listing 21: Joc de Prova 1 - Extensió 1

```
(define (problem
    redflix-problem)
(:domain redflix)

(:objects
bb_s1 bb_s2 bb_s3 bb_s4 bb_s5
    cc1 cc2 w1 - content
day1 day2 day3 day4 day5 -
    day)

(:init
(watched bb_s4)
(watched bb_s3)
(watched w1)

(is_wanted bb_s5)
(is_wanted cc2)

(predecessor bb_s1 bb_s5)
(predecessor bb_s2 bb_s5)
(predecessor bb_s3 bb_s5)
(predecessor bb_s4 bb_s5)

(yesterday day1 day2)
(yesterday day2 day3)
(yesterday day3 day4)
(yesterday day4 day5))

(:goal (and
(assigned bb_s1)
(assigned bb_s2)
(assigned cc2)))
)
```

Listing 22: Joc de Prova 2 - Extensió 1

C.4 Extensió 2

Joc de prova 1

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint diversos continguts predecessors i paral·lels els continguts desitjats, afegint-los al pla també complint amb les restriccions de precedència. Veure que es poden assignar continguts sense límit a un dia.

Entrada: Tenim 8 continguts en total, es desitgen veure 1 continguts. Hi ha 5 dies disponibles. Un contingut té 4 continguts predecessors en cadena i 3 continguts paral·lels. En el *listing* d'avall es pot veure el codi del problema. detalladament(23)

Sortida esperada: S'espera que el planificador assigni els continguts predecessors en cadena en els 5 dies disponibles i que els paral·lels al contingut desitjat un dia abans o el mateix dia.

Sortida obtinguda: Les restriccions de precedència i paral·lelisme es compleixen. Estan afegits al pla tots els continguts desitjats amb els seus predecessors i paral·lels. Veiem que al dia anterior al desitjat s'afegeixen tots els paral·lels sense límit. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: BB_S1
DAY2: BB_S2
DAY3: BB_S3
DAY4: BB_S4, AA1, AA2, AA3
DAY5: BB_S5
```

Figura 5: Sortida de FFInterpreter.py (B.1) del E2-J1.

Joc de prova 2

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint diversos continguts paral·lels els continguts desitjats, afegint-los al pla també complint amb les restriccions de precedència, sense que s'afegeixin els que ja s'han vist.

Entrada: Tenim 10 continguts totals, dels quals només es vol veure un, 2 ja s'han vist, els quals són paral·lels del contingut desitjat. La resta de continguts son paral·lels al contingut desitjat. Hi ha 2 dies disponibles. En el *listing* d'avall es pot veure el codi del problema detalladament(24)

Sortida esperada: El planificador hauria d'assignar en un dia el contingut desitjat i entre el dia anterior i el mateix dia tots els seus paral·lels que no s'han vist encara.

Sortida obtinguda: Les restriccions de paral·lelisme es compleix, no s'assignen els que ja s'han vist i s'han assignat al primer dia els que son paral·lels i el segon dia el desitjat. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
DAY1: AA1, AA2, AA3, CC1, BB_S3, BB_S4, BB_S5
DAY2: CC2
```

Figura 6: Sortida de FFInterpreter.py (B.1) del E2-J2.

```
(define (problem
    redflix-problem)
(:domain redflix)

(:objects
bb_s1 bb_s2 bb_s3 bb_s4 bb_s5
    aa1 aa2 aa3 - content
day1 day2 day3 day4 day5 -
    day)

(:init
(is_wanted bb_s5)

;;predecesores
(predecessor bb_s1 bb_s2)
(predecessor bb_s2 bb_s3)
(predecessor bb_s3 bb_s4)
(predecessor bb_s4 bb_s5)

;;paralelos
(parallel aa1 bb_s5)
(parallel aa2 bb_s5)
(parallel aa3 bb_s5)

;; Dias anteriores
(yesterday day1 day2)
(yesterday day2 day3)
(yesterday day3 day4)
(yesterday day4 day5))

(:goal (and
(day_to_watch bb_s1 day1)
(assigned aa1)
(assigned aa2)
(assigned aa3)))
)
```

Listing 23: Joc de Prova 1 - Extensió 2

```
(define (problem
    redflix-problem)
(:domain redflix)

(:objects
bb_s1 bb_s2 bb_s3 bb_s4 bb_s5
    aa1 aa2 aa3 cc1 cc2 -
    content
day1 day2 - day)

(:init
(watched bb_s1)
(watched bb_s2)
(is_wanted cc2)

(parallel cc1 cc2)
(parallel aa1 cc2)
(parallel aa2 cc2)
(parallel aa3 cc2)
(parallel bb_s5 cc2)
(parallel bb_s4 cc2)
(parallel bb_s3 cc2)
(parallel bb_s2 cc2)
(parallel bb_s1 cc2)

(yesterday day1 day2))

(:goal (and
(assigned aa1)
(assigned aa2)
(assigned aa3)
(assigned cc1)
(assigned bb_s3)
(assigned bb_s4)
(assigned bb_s5)))
)
```

Listing 24: Joc de Prova 2 - Extensió 2

C.5 Extensió 3

Joc de prova 1

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint més d'un contingut predecessor i paral·lel als continguts desitjats per veure, afegint-los al pla també i que es respectin les restriccions de visualització. I comprovar que no s'afegeixin al pla continguts ja vists que siguin predecessors als desitjats i tampoc els següents dels desitjats.

Entrada: Tenim 13 continguts en total, es desitgen veure bb_s5, bb_s3, aa2, cc1, cc2 i cc3, i s'han vist bb_s4 i aa1. Hi ha només 3 dies disponibles. El contingut bb_s5 és predecessor d'un contingut i paral·lel d'un altre; té 3 continguts predecessors i un d'aquests un altre predecessors; i a més 2 paral·lels. En el *listing* d'avall es pot veure el codi del problema. detalladament(25)

Sortida esperada: S'espera que el planificador assigni per veure 9 dels 13 continguts totals distribuïts entre els 3 dies. Al dia 3 el contingut bb_s5 i el seu paral·lel ja que el dia anterior s'ha d'assignar els seus 3 predecessors i a l'anterior l'altre predecessor. I finalment els altres 3 continguts distribuïts en els dies on encara hi hagi lloc.

Sortida obtinguda: Les restriccions de continguts per dia es respecten i també els continguts paral·lels i predecessors es veuen quan toca. Estan afegits al pla tots els continguts desitjats amb els seus predecessors i paral·lels. No estan afegits els continguts que ja s'han vist i són predecessors i tampoc els següents del

que es vol veure. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: CC3, BB_S1, BB_S0
DAY2: CC2, BB_S3, BB_S2
DAY3: CC1, BB_S5, AA2
```

Figura 7: Sortida de FFInterpreter.py (B.1) del E3-J1.

Joc de prova 2

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint continguts predecessors i paral·lels als continguts desitjats per veure, afegint-los al pla també i que es respectin les restriccions de visualització.

Entrada: Tenim 5 continguts, cadascun predecessor d'un altre en cadena on el que no és predecessor de cap es vol veure (bb_s5). També tenim 3 continguts paral·lels a l'anterior que es vol veure i dos continguts més on un es vol veure i l'altre és predecessor d'aquest. Hi ha 5 dies disponibles per assignar els continguts. En el *listing* d'avall es pot veure el codi del problema detalladament(26)

Sortida esperada: El planificador hauria d'assignar els 5 continguts bb_sx un a cada dia en ordre ascendent ja que són predecessors en cadena, assignant al dia 5 el contingut bb_s5. Els paral·lels a bb_s5 haurien d'estar entre el dia 4 i 5, respectant el màxim de 3 continguts per dia. Per últim els dos altres continguts en dies consecutius respectant lo del predecessor.

Sortida obtinguda: Les restriccions de continguts per dia es respecten i també els continguts paral·lels i predecessors es veuen quan toca. Estan afegits al pla tots els continguts desitjats amb els seus predecessors i paral·lels. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: CC1, BB_S1
DAY2: CC2, BB_S2
DAY3: BB_S3
DAY4: BB_S4, AA3
DAY5: BB_S5, AA1, AA2
```

Figura 8: Sortida de FFInterpreter.py (B.1) del E3-J2.

```
(define (problem redflix-problem)
(:domain redflix)

(:objects
  bb_s0 bb_s1 bb_s2 bb_s3 bb_s4
      bb_s5 bb_s6 bb_s7 aa1 aa2
      cc1 cc2 cc3 - content
  day1 day2 day3  - day)

(:init
  ;; Contenidos ya vistos
  (watched bb_s4)
  (watched aa1)

  ;; Contenidos por ver
  (is_wanted bb_s5)
  (is_wanted bb_s3)
  (is_wanted aa2)
  (is_wanted cc1)
  (is_wanted cc2)
  (is_wanted cc3)

  ;; Relaciones entre contenidos
      predecesores
  (predecessor bb_s0 bb_s2)
  (predecessor bb_s1 bb_s2)
  (predecessor bb_s2 bb_s5)
  (predecessor bb_s3 bb_s5)
  (predecessor bb_s4 bb_s5)
  (predecessor bb_s5 bb_s6)

  ;;paralelos
  (parallel aa1 bb_s5)
  (parallel aa2 bb_s5)
  (parallel bb_s5  bb_s7)

  ;; Dias anteriores
  (yesterday day1 day2)
  (yesterday day2 day3))

(:goal (and
  (assigned bb_s1)
  (assigned bb_s0)
  (assigned bb_s3)
  (assigned aa2)
  (assigned cc1)
  (assigned cc2)
  (assigned cc3)))
)
```

Listing 25: Joc de Prova 1 - Extensió 3

```
(:domain redflix)

(:objects
  bb_s1 bb_s2 bb_s3 bb_s4 bb_s5
      aa1 aa2 aa3 cc1 cc2 -
      content
  day1 day2 day3 day4 day5 day6
      day7 - day)

(:init
  (is_wanted bb_s5)
  (is_wanted cc2)

  (predecessor bb_s1 bb_s2)
  (predecessor bb_s2 bb_s3)
  (predecessor bb_s3 bb_s4)
  (predecessor bb_s4 bb_s5)
  (predecessor cc1 cc2)

  (parallel aa1 bb_s5)
  (parallel aa2 bb_s5)
  (parallel aa3 bb_s5)

  (yesterday day1 day2)
  (yesterday day2 day3)
  (yesterday day3 day4)
  (yesterday day4 day5))

(:goal
  (and
  (day_to_watch bb_s1 day1)
  (assigned aa1)
  (assigned aa2)
  (assigned aa3)
  (assigned cc1)))
)
```

Listing 26: Joc de Prova 2 - Extensió 3

Joc de prova Random

Objectiu: Aquest joc de prova està fet pel generador de problemes i es vol comprovar que també és capaç de planificar un problema generat amb configuracions aleatòries.

Entrada: Tenim 21 continguts, 5 dels quals són predecessors d'un altre en cadena que són els que interessen ja que els últims de la cadena es desitgen veure. Un altre contingut apart es desitja veure. Un d'aquests té un contingut paral·lel. Després hi ha definides més relacions però que no son rellevants. Hi ha 9 dies disponibles per assignar els continguts. En el *listing* d'avall es pot veure el codi del problema sencer detalladament(27)

Sortida esperada: S'espera que el planificador assigni els 5 continguts predecessors en cadena en 5 dies consecutius ja que son predecessors del contingut desitjat. Que el paral·lel a un d'aquests continguts predecessors estigui en el mateix dia o el anterior i l'últim contingut que queda en qualsevol que càpiga.

Sortida obtinguda: Les restriccions de continguts per dia es respecten i també els continguts paral·lels i predecessors es veuen quan toca. El planificador ha trobat un pla que respecta les restriccions d'un problema aleatori. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: C10, C1
DAY2: C2
DAY3: C3, C18
DAY4: C4
DAY5: C5
```

Figura 9: Sortida de FFInterpreter.py (B.1) del E3-RND.

```
(define (problem random-redflix-problem)
(:domain redflix)

(:objects
  y1 day2 day3 day4 day5 day6 day7 day8 day9 - day)

(:init
  (predecessor c1 c2)
  (predecessor c2 c3)
  (predecessor c3 c4)
  (predecessor c4 c5)
  (predecessor c12 c13)
  (predecessor c13 c14)
  (predecessor c16 c17)
  (predecessor w1 c1)

  (parallel c9 c12)
  (parallel c18 c3)
  (parallel c15 c9)
  (parallel w2 c12)
  (parallel w3 c12)

  (is_wanted c4)
  (is_wanted c10)
  (is_wanted c5)

  (watched w1)
  (watched w2)
  (watched w3)

  (yesterday day1 day2)
  (yesterday day2 day3)
  (yesterday day3 day4)
  (yesterday day4 day5)
  (yesterday day5 day6)
  (yesterday day6 day7)
  (yesterday day7 day8)
  (yesterday day8 day9))

```

```
  (:goal
    (and
    (day_to_watch c1 day1)
    (assigned c18)
    (assigned c10)))
  )
```

Listing 27: Joc de Prova Random - Extensió 3

C.6 Extensió 4

Joc de prova 1

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint més d'un contingut predecessor en cadena i paral·lels als continguts desitjats per veure, afegint-los al pla també i que es respectin les restriccions de visualització. Observar que no s'assignin continguts a més d'un dia.

Entrada: Tenim 17 continguts en total, es desitgen veure 2. Hi ha només 12 dies disponibles. Un d'aquests continguts té un predecessor i l'altre té 3 paral·ls i 11 predecessors en cadena. Tots tenen una duració entre 50 i 100 minuts. En el *listing* d'avall es pot veure el codi del problema. detalladament(28)

Sortida esperada: S'espera que el planificador assigni per veure tots els continguts totals distribuïts entre tots els dies. Entre els 12 dies s'han d'assignar els 12 continguts predecessors en cadena i en els dos últims dies els paral·lels al contingut desitjat. I els altres dos continguts predecessors en dies consecutius lo més propers de dies.

Sortida obtinguda: Les restriccions de temps per dia es respecten i també els continguts paral·lels i predecessors es veuen quan toca. Els continguts cc1 i cc2 s'han assignat als dies més pròxims en els dies 1 i 2. Estan afegits al pla tots els continguts desitjats amb els seus predecessors i paral·lels. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: CC1, BB_S1
DAY10: BB_S10
DAY11: BB_S11, AA1, AA3
DAY12: BB_S12, AA2
DAY2: CC2, BB_S2
DAY3: BB_S3
DAY4: BB_S4
DAY5: BB_S5
DAY6: BB_S6
DAY7: BB_S7
DAY8: BB_S8
DAY9: BB_S9
```

Figura 10: Sortida de FFInterpreter.py (B.1) del E4-J1.

```
(define (problem redflix-problem)
(:domain redflix)

(:objects
  bb_s1 bb_s2 bb_s3 bb_s4 bb_s5 bb_s6 bb_s7 bb_s8 bb_s9 bb_s10 bb_s11 bb_s12
      aa1 aa2 aa3 cc1 cc2 - content
  day1 day2 day3 day4 day5 day6 day7 day8 day9 day10 day11 day12 - day)

(:init
```

```
    (is_wanted bb_s12)  (is_wanted cc2)
    (predecessor bb_s1 bb_s2)
    (predecessor bb_s2 bb_s3)
    (predecessor bb_s3 bb_s4)
    (predecessor bb_s4 bb_s5)
    (predecessor bb_s5 bb_s6)
    (predecessor bb_s6 bb_s7)
    (predecessor bb_s7 bb_s8)
    (predecessor bb_s8 bb_s9)
    (predecessor bb_s9 bb_s10)
    (predecessor bb_s10 bb_s11)
    (predecessor bb_s11 bb_s12)
    (predecessor cc1 cc2)
    (parallel aa1 bb_s12)
    (parallel aa2 bb_s12)
    (parallel aa3 bb_s12)
    (yesterday day1 day2)
    (yesterday day2 day3)
    (yesterday day3 day4)
    (yesterday day4 day5)
    (yesterday day5 day6)
    (yesterday day6 day7)
    (yesterday day7 day8)
    (yesterday day8 day9)
    (yesterday day9 day10)
    (yesterday day10 day11)
    (yesterday day11 day12)
    (= (duration bb_s1) 50)
    (= (duration bb_s2) 70)
    (= (duration bb_s3) 80)
    (= (duration bb_s4) 50)
    (= (duration bb_s5) 70)
    (= (duration bb_s6) 80)
    (= (duration bb_s7) 50)
    (= (duration bb_s8) 70)
    (= (duration bb_s9) 80)
    (= (duration bb_s10) 50)
    (= (duration bb_s11) 70)
    (= (duration bb_s12) 90)
    (= (duration cc1) 60)
    (= (duration cc2) 100)
    (= (duration aa1) 50)
    (= (duration aa2) 70)
    (= (duration aa3) 80)
    (= (day_duration day1) 0)
    (= (day_duration day2) 0)
    (= (day_duration day3) 0)
    (= (day_duration day4) 0)
    (= (day_duration day5) 0)
    (= (day_duration day6) 0)
    (= (day_duration day7) 0)
    (= (day_duration day8) 0)
    (= (day_duration day9) 0)
    (= (day_duration day10) 0)
    (= (day_duration day11) 0)
    (= (day_duration day12) 0)
    (= (priority day1) 1) (= (priority day2) 2)
    (= (priority day3) 3)   (= (priority day4) 4)
    (= (priority day5) 5)   (= (priority day6) 6)
    (= (priority day7) 7)   (= (priority day8) 8)
    (= (priority day9) 9)   (= (priority day10) 10)
    (= (priority day11) 11) (= (priority day12) 12)
    (= (total-days) 0)
```

```
    (= (remaining-content) 17))

  (:goal (and
    (= (remaining-content) 0)))
  (:metric minimize (total-days))
  )
```

Listing 28: Joc de Prova 1 - Extensió 4

Joc de prova 2

Objectiu: Aquest joc de prova està dissenyat per avaluar si el planificador és capaç de planificar correctament tenint continguts predecessors i paral·lels als continguts desitjats per veure, afegint-los al pla també, sense afegir els que ja s'han vist. Fent èmfasi en el compliment del màxim de minuts per dia i en la minimització de dies i que siguin els primers.

Entrada: Tenim 16 continguts en total, dels quals es volen veure 2. Un dels que es desitja té un predecessor i aquest predecessor té 7 predecessors més, els quals tenen una duració baixa de 20 minuts i un ja s'ha vist, i 3 paral·lels amb 30 minuts de duració. Hi ha 10 dies disponibles. En el *listing* d'avall es pot veure el codi del problema detalladament(29)

Sortida esperada: El planificador hauria d'assignar el contingut desitjat al tercer dia, el seu predecessor el dia anterior juntament amb els seus paral·lels i els 6 continguts predecessors no vists en un mateix dia el primer dia, minimitzant el dies així.

Sortida obtinguda: Les restriccions de continguts per dia es respecten i també els continguts paral·lels i predecessors es veuen quan toca. El primer dia s'han assignat el màxim de continguts possibles on estan tots els predecessors i algun paral·lel.Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: C2, C1, A1, A2, A3, A5, A6, A7
DAY2: B1, C3, C4
DAY3: B2
```

Figura 11: Sortida de FFInterpreter.py (B.1) del E4-J2.

```
(define (problem redflix-problem)
(:domain redflix)

(:objects
  a1 a2 a3 a4 a5 a6 a7 b1 b2 b3 c1 c2 c3 c4 d1 d2 - content
  day1 day2 day3 day4 day5 day6 day7 day8 day9 day10 - day)

(:init
  (watched a4)
  (watched d1)
  ;; Contenidos por ver
  (is_wanted b2)
  (is_wanted c2)

  (predecessor a1 b1)
  (predecessor a2 b1)
  (predecessor a3 b1)
  (predecessor a4 b1)
  (predecessor a5 b1)
  (predecessor a6 b1)
  (predecessor a7 b1)
```

```
    (predecessor b1 b2)
    (predecessor b2 d2)

    ;;paralelos
    (parallel c1 b1)
    (parallel c2 b1)
    (parallel c3 b1)
    (parallel c4 b1)
    ;; Dias anteriores
    (yesterday day1 day2)
    (yesterday day2 day3)
    (yesterday day3 day4)
    (yesterday day4 day5)
    (yesterday day5 day6)
    (yesterday day6 day7)
    (yesterday day7 day8)
    (yesterday day8 day9)
    (yesterday day9 day10)

    (= (duration a1) 20)
    (= (duration a2) 20)
    (= (duration a3) 20)
    (= (duration a4) 20)
    (= (duration a5) 20)
    (= (duration a6) 20)
    (= (duration a7) 20)
    (= (duration b1) 70)
    (= (duration b2) 80)
    (= (duration b3) 50)
    (= (duration c1) 30)
    (= (duration c2) 30)
    (= (duration c3) 30)
    (= (duration c4) 30)
    (= (duration d1) 60)
    (= (duration d2) 60)
    (= (day_duration day1) 0)
    (= (day_duration day2) 0)
    (= (day_duration day3) 0)
    (= (day_duration day4) 0)
    (= (day_duration day5) 0)
    (= (day_duration day6) 0)
    (= (day_duration day7) 0)
    (= (day_duration day8) 0)
    (= (day_duration day9) 0)
    (= (day_duration day10) 0)

    (= (priority day1) 1)
    (= (priority day2) 2)
    (= (priority day3) 3)
    (= (priority day4) 4)
    (= (priority day5) 5)
    (= (priority day6) 6)
    (= (priority day7) 7)
    (= (priority day8) 8)
    (= (priority day9) 9)
    (= (priority day10) 10)

    (= (total-days) 0)
    (= (remaining-content) 12)

  (:goal (and
    (= (remaining-content) 0)))
  (:metric minimize (total-days))
```

```
)
```

Listing 29: Joc de Prova 2 - Extensió 4

Joc de prova Random

Objectiu: Aquest joc de prova està fet pel generador de problemes i es vol comprovar que també és capaç de planificar un problema generat amb configuracions aleatòries i assigni els continguts als dies més propers possible complint les restriccions de temps per dia.

Entrada: Tenim 20 continguts, dels quals 9 son desitjats de veure. Entre ells hi ha establertes relacions de predecessors i paral·lels i només un apart és predecessor d'un dels desitjats. Hi ha un paral·lel a un contingut desitjat però ja s'ha vist. Hi ha 14 dies disponibles. Les duracions dels dies han estat assignades de manera aleatòria per lo que està variat. En el *listing* d'avall es pot veure el codi del problema sencer detalladament(30)

Sortida esperada: S'espera que el planificador assigni els 4 continguts predecessors en cadena que hi ha, que són desitjats tots menys un, en 4 dies consecutius. Que el parell de paral·lels i el de consecutius més estigui en el mateix dia o en consecutius segons pertoqui; i, els últims continguts que queda en qualsevol que càpiga. Com a molt ocupi 6 dies i que siguin els primers 6 dies.

Sortida obtinguda: Les restriccions de temps per dia es respecten i també els continguts paral·lels i predecessors es veuen quan toca. El planificador ha trobat un pla que respecta les restriccions d'un problema aleatori. Podem veure a continuació el resultat del pla que ha trobat dia per dia i coincideix amb el que s'esperava:

```
Planificación por día:

DAY1: C11, C14
DAY2: C7, C8
DAY3: C9, C1
DAY4: C17, C2
DAY5: C3
DAY6: C4
```

Figura 12: Sortida de FFInterpreter.py (B.1) del E4-RND.

```
(define (problem random-redflix-problem)
(:domain redflix)

(:objects
  c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 w1 w2 w3 -
      content
  day1 day2 day3 day4 day5 day6 day7 day8 day9 day10 day11 day12 day13 day14
      - day)

(:init
  (predecessor c1 c2)
  (predecessor c2 c3)
  (predecessor c3 c4)
  (predecessor c4 c5)
  (predecessor c8 c9)
  (predecessor c11 c12)
  (predecessor c12 c13)
  (predecessor c15 c16)
  (predecessor w1 c5)

```

```
    (parallel c7 c6)
    (parallel c14 c7)
    (parallel w3 c4)

    (is_wanted c4)
    (is_wanted c9)
    (is_wanted c11)
    (is_wanted c17)
    (is_wanted c7)
    (is_wanted c14)
    (is_wanted c2)
    (is_wanted c3)
    (is_wanted c8)

    (watched w1)
    (watched w2)
    (watched w3)

    (= (duration c1) 87)
    (= (duration c2) 58)
    (= (duration c3) 59)
    (= (duration c4) 90)
    (= (duration c5) 66)
    (= (duration c6) 52)
    (= (duration c7) 65)
    (= (duration c8) 97)
    (= (duration c9) 70)
    (= (duration c10) 111)
    (= (duration c11) 78)
    (= (duration c12) 83)
    (= (duration c13) 112)
    (= (duration c14) 91)
    (= (duration c15) 68)
    (= (duration c16) 44)
    (= (duration c17) 102)
    (= (duration w1) 102)
    (= (duration w2) 52)
    (= (duration w3) 75)

    (yesterday day1 day2)
    (yesterday day2 day3)
    (yesterday day3 day4)
    (yesterday day4 day5)
    (yesterday day5 day6)
    (yesterday day6 day7)
    (yesterday day7 day8)
    (yesterday day8 day9)
    (yesterday day9 day10)
    (yesterday day10 day11)
    (yesterday day11 day12)
    (yesterday day12 day13)
    (yesterday day13 day14)

    (= (day_duration day1) 0)
    (= (day_duration day2) 0)
    (= (day_duration day3) 0)
    (= (day_duration day4) 0)
    (= (day_duration day5) 0)
    (= (day_duration day6) 0)
    (= (day_duration day7) 0)
    (= (day_duration day8) 0)
    (= (day_duration day9) 0)
    (= (day_duration day10) 0)
```

```
    (= (day_duration day11) 0)
    (= (day_duration day12) 0)
    (= (day_duration day13) 0)
    (= (day_duration day14) 0)

    (= (priority day1) 1)
    (= (priority day2) 2)
    (= (priority day3) 3)
    (= (priority day4) 4)
    (= (priority day5) 5)
    (= (priority day6) 6)
    (= (priority day7) 7)
    (= (priority day8) 8)
    (= (priority day9) 9)
    (= (priority day10) 10)
    (= (priority day11) 11)
    (= (priority day12) 12)
    (= (priority day13) 13)
    (= (priority day14) 14)

    (= (total-days) 0)
    (= (remaining-content) 10))

  (:goal
    (and
    (= (remaining-content) 0)))

  (:metric minimize (total-days))
  )
```

Listing 30: Joc de Prova Random - Extensió 4