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, complint amb restriccions com el respecte als predecedents i continguts paral·lels.

La pràctica s'ha de desenvolupar amb el planificador *Fast Forward* v2.3 i inclou diverses extensions que permeten incrementar la dificultat i el valor de l'activitat. 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, afegint restriccions al nombre d'activitats diàries.
- **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.

Aquestes extensions ofereixen als estudiants la possibilitat d'abordar reptes més avançats i de demostrar una comprensió més profunda de la planificació en espai d'estats, així com de millorar el seu domini del llenguatge PDDL. Cada extensió també es tradueix en un increment potencial de la nota màxima de la pràctica, incentivant el desenvolupament de solucions més completes i sofisticades.

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.

2.1 Variables

Les variables que representen el nostre domini son dues. Aquestes són:

- `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 conforme avancem per les diferents extensions.

- `(watched ?c - content)`: Indica que un contingut `?c` ja s'ha vist.
- `(is_wanted ?c - content)`: Indica que l'usuari pot veure un contingut `?c`.
- `(predecessor ?c1 - content ?c2 - content)`: Indica que el contingut `?c1` és predecessor de `?c2`. Per tant, `?c1` s'ha de veure abans que `?c2`.
- `(day_to_watch ?c - content ?d - day)`: Assigna un contingut `?c` a un dia `?d`.
- `(yesterday ?d1 - day ?d2 - day)`: Indica que `?d1` és el dia anterior a `?d2`.
- `(assigned ?c - content)`: Marca que un contingut `?c` ja té un dia assignat.

Siguin aquests els predicats del **Nivell bàsic**, ara ennumerarem 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 A B)`
  `(parallel B A)`

Extensió 3: Apareixen tres predicats nous, cada un indicant quin dels tres continguts de cada dia han sigut ja assignats. Aquestes 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 la 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:

- `(total-days)`: Enregistra el nombre total de dies utilitzats. Necessari per optimitzar el pla.
- `(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. Per explicar-les farem una breu desprescripció del seu funcionament i explicarem les diferents iteracions que han tingut al llarg de les diferents 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 son diferents, analitzarem en que canvien.

Les 3 accions són les següents:

add_content

Aquesta acció afegeix continguts que l'usuari vol veure o que el sistema vol que l'usuari vegi.

- **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.

- **Precondicions**: El contingut és desitjat, no s'ha vist, no té successors ni paral·lels sense assignar, i el dia té espai disponible.
- **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))
      (or
        (not (assigned_one ?d))
        (not (assigned_two ?d))
        (not (assigned_three ?d)))
      ;; Verifica que no haya sucesores que no hayan sido vistos
      (not (exists (?c2 - content)
        (and
          (predecessor ?c ?c2)
          (not (watched ?c2)))))
      ;; Verifica que no haya contenidos paralelos que no hayan sido vistos
      (not (exists (?c2 - content)
        (and
          (parallel ?c ?c2)
          (not (watched ?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))
  ))
```

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. Després per l'**Extensió 3**, s'afegeixen les línies de la precondició `or` de `(not (assigned_<number> ?d))`, i les de la postcondició també..

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) 1)`: incrementa els dies usats.

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

set_day

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

- **Precondicions**: El contingut és desitjat, compleix amb les relacions de predecessors i paral·lels, i el dia té espai disponible.
- **Efectes**: Assigna el contingut al dia seleccionat, respectant les restriccions.

```
;; 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.
(:action set_day
  :parameters (?c1 ?c2 - content ?d1 ?d2 - day)
  :precondition (and
    (is_wanted ?c1)
    (day_to_watch ?c2 ?d2)
```

```
      (or
        (not (assigned_one ?d1))
        (not (assigned_two ?d1))
        (not (assigned_three ?d1)))
      (or
        ;condiciones para contenido paralelo
        (and
          (parallel ?c1 ?c2)
          (or
            (previous ?d1 ?d2)
            (= ?d1 ?d2)))
        ;condiciones para contenido predecessor
        (and
          (predecessor ?c1 ?c2)
          (previous ?d1 ?d2))))
    :effect (and
      (day_to_watch ?c1 ?d1)
      (assigned ?c1)
      (when (not (assigned_one ?d1)) (assigned_one ?d1))
      (when (and (assigned_one ?d1) (not (assigned_two ?d1))) (assigned_two
          ?d1))
      (when (and (assigned_one ?d1) (assigned_two ?d1) (not (assigned_three
          ?d1))) (assigned_three ?d1))
  ))
```

Listing 3: Acció set_day

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

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) 1)`: incrementa els dies usats.

`(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 cuatre dies, es definirien així:

```
(:objects
  bb_s1 bb_s2 bb_s3 cc1 cc2 - 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:

```
(: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: (continguts predecessors, paral·lels, etc.).

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

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.

4.1 Nivell Bàsic

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

4.2 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 (7). 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 7: Fragment per assignar més d'un predecessor com a pendent de veure

4.3 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 8: 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)))
)
```

Listing 9: 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
  (parallel cc1 cc2)
  (yesterday day1 day2)
```

```
    (yesterday day2 day3)
  )
```

Listing 10: Inicialització de paral·lels i dies consecutius

4.4 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`. 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 11: 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.

4.5 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)

    ;; POSTCONDICIO
    (increase (day_duration ?d) (duration ?c))
```

Listing 12: 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:

```
(:functions
  (duration ?c - content)
  (day_duration ?d - day)
)
```

Listing 13: 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. Per exemple:

```
(:init
  (= (duration bb_s1) 45)
  (= (duration bb_s2) 50)
  (= (duration bb_s3) 60)
  (= (duration cc1) 30)
  (= (duration cc2) 25)

  (= (day_duration day1) 0)
  (= (day_duration day2) 0)
  (= (day_duration day3) 0)
)
```

Listing 14: Novetats en la inicialització per l'extensió 4

5 Jocs de prova

**********El conjunto de problemas de prueba (mínimo 2 por extensión), explicando para cada uno que es lo que intentan probar y su resultado. Podéis partir de los juegos de pruebapara el nivel básico e ir añadiendo los elementos que cada extensión requiera. Si habéisimplementado el generador de problemas, al menos uno de los juegos de prueba de cadaextensión ha de ser obtenido de este.

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.

5.1 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 es igual sempre.

5.2 Nivell Bàsic

5.3 Extensió 1

5.4 Extensió 2

5.5 Extensió 3

5.6 Extensió 4

6 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 significatiu, 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.

6.1 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 a identificar possibles limitacions, fet que ens ha ajudat a entendre millor les capacitats i les limitacions dels sistemes de planificació.

6.2 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.

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

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 15: 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 16: Especificació del problema en PDDL

B Annexos

B.1 Intèrpret per a sortides de FF i MFF

Aquest annex descriu l'algorisme utilitzat per analitzar les sortides generades pels planificadors Fast Forward (FF) i Metric Fast Forward (MFF). S'ha fet mitjançant Python.

Aquest programa està dissenyat per interpretar la sortida generada pels planificadors automàtics Fast Forward (FF) i Metric Fast Forward (MFF). En iniciar-se, el programa presenta diverses opcions a l'usuari, com triar entre FF (Extensió 3), MFF amb un problema generat aleatòriament, o MFF amb un problema específic (Extensió 4). Un cop seleccionada l'opció, el programa executa el planificador corresponent i captura la seva sortida. Aquesta sortida és analitzada per extreure les accions rellevants, com ara assignacions de contingut o configuracions de dies únics. Finalment, les accions són organitzades en una estructura que mostra les tasques agrupades per dia, facilitant la seva visualització i interpretació com una taula o llista estructurada. Això permet obtenir una visió clara del pla generat pel sistema.

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`