Universidad Politécnica de Catalunya

Inteligencia Artificial

Planificación

Planificador de Series y Películas

Autores
Pau Celma
Jazmin E. Gell
Ramon Zalabardo

Supervisor
Javier Béjar Alonso

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Enero ◇ QT - 2020/21

Contents

1 Introducción

Esta es la documentación de la práctica de planificación de la asignatura de Inteligencia Artificial, en el cuatrimestre de otoño del curso 20/21. Se nos pide desarrollar en Fast Forward v2.3 un planificador de contenidos audiovisuales en base a las necesidades de varios tipos de usuario.

En este proyecto, queremos poder describir e implementar el dominio que se plantea conjuntamente con distintos ejemplos de problemas a través de lenguajes de descripción (PDDL). De manera que podamos familiarizarnos con lenguajes de representación de acciones y se puedan aplicar planificadores modernos para conseguir inferir una solución que cumpla los requisitos que se nos piden.

2 Dominio

2.1 Estado inicial y final

Para todos los problemas el estado final es el mismo, es un estado en el que no existan contenidos por ver que no se hayan visto. Sin embargo el estado inicial es distinto para cada uno de los problemas.

Nivel básico: En el nivel básico se empieza con un estado inicial en el que solo se tiene la información de los predecesores de cada contenido y los contenidos que se quieren visionar.

Extensión 1: En la extensión 1 se empieza con un estado inicial en el que además de conocer la información de los predecesores y de los contenidos que se quieren visionar, también se conoce para cada día cuales son los días anteriores a este.

Extensión 2: En la extensión 2 se conoce la misma información que en la extensión 1 y además contamos con la información sobre el orden concreto de los días.

Extensión 3: En la extensión 3 además de conocerse la misma información que en la extensión 2 también se inicia con una variable *contenidosDia* que contiene el valor 0.

Extensión 4: En la extensión 4 además de conocerse la misma información que en la extensión 2 también se inicia con dos variables *minContenido* que contiene para cada contenido los minutos que este dura y *minLibresDia* que contiene el valor 200.

2.2 Variables

2.2.1 Contenido

Es tipo object y representa el contenido audiovisual. Es el principal objeto de nuestro dominio, pues será el que tiene que ser repartido a lo largo del tiempo. Este puede estar relacionado con otros contenidos de manera que sean predecesores o paralelos. Es decir, que para poder verlo primero se deben haber visto sus predecesores en días anteriores. Los paralelos se deben ver el mismo día o uno antes (o después por simetría). En la expansión 4 se dota al contenido de la propiedad de duración: cada contenido tiene asignado un tiempo en minutos. El contenido puede haber sido ya visto por el usuario y puede querer ser visto.

2.2.2 Día

Es tipo object y representa la medida de tiempo en que se reagrupan los contenidos. Los días tienen una relación de orden, es decir, pueden tener anteriores y posteriores. También, en la expansión 4 se les dota de una capacidad de 200 minutos. El tiempo máximo que se puede usar para ver serie/películas, diferente a las versiones previas en que un día podía contener indistintamente contenido o en la expansión 3 un máximo de 3 contenidos distintos. La cantidad de días es el número de días que el usuario tiene disponibles para poder planificar las series que quiere ver.

2.3 Predicados

Para modelar los problemas, hemos ideado los siguientes predicados:

- *visto(Contenido)*: Indica que ya se ha añadido un contenido al plan de visionado. Nos es útil para saber si podemos añadir un contenido al plan.
- *vistoDia(Contenido, Día)*: Indica que ya se ha añadido un contenido al plan de visionado e indica a qué día se ha asignado. Nos es útil para saber si el predecesor o paralelo de un contenido está en un día válido respecto a donde queremos insertar el contenido.
- *visionado(Contenido)*: Indica que el usuario quiere tener ese contenido en el plan de visionado. Es uno de los predicados que define las condiciones iniciales de los problemas, y nos ayuda a generalizar la inserción de contenido de cara a añadir predecesores y paralelos al plan.
- *predecesor(Contenido, Contenido)*: Indica que el primer contenido precede al segundo. Nos es útil para asegurarnos de que el programa cumple con la restricción de que un contenido predecesor debe verse en un día anterior al de su sucesor.
- *paralelo(Contenido, Contenido)*: Indica que el primero contenido es paralelo al segundo. Nos es útil para asegurarnos de que el programa cumple con la restricción de que dos contenidos paralelos se deben ver con, como mucho, un intervalo de un día entre ellos.
- *antes(Día, Día)*: Indica que el primer día es inmediatamente anterior al segundo día. Nos es útil para saber a qué día podemos asignar un contenido paralelo a otro.
- *anterior(Día, Día)*: Indica que el primer día es anterior al segundo día. Nos es útil para saber a qué día podemos asignar un contenido predecesor de otro.

2.4 Funciones

2.4.1 contenidosDia

Exclusiva de la extensión 3, la función *contenidosDia* indica cuantos contenidos se han asignado a un día en particular en el plan de visionado. Nos sirve para cumplir la restricción de la extensión 3, que exige que ningún día tenga 3 o más contenidos asignados.

2.4.2 minLibresDia

Exclusiva de la extensión 4, la función *minLibresDia* indica los minutos sin contenido asignado que tiene un día en particular. Nos sirve para cumplir la restricción de la extensión 4, que exige que ningún día tenga más de 200 minutos de contenido asignados.

2.4.3 minContenido

Exclusiva de la extensión 4, la función *minContenido* indica los minutos de duración de un contenido. Nos sirve para calcular los minutos restantes en un día tras asignarle un contenido en el plan de visionado.

2.5 Operadores

2.5.1 Añadir predecesores

El operador de añadir predecesores se usa para asegurarnos de que un contenido predecesor a otro se verá en un día anterior a su sucesor. La idea detrás de este operador es añadir el predecesor de un contenido a la lista de contenido que se debe incluir en el plan de visionado. El contenido sucesor deberá estar marcado para ser añadido al plan,

pero sin haber sido añadido aún. Hacemos esto sabiendo que el operador encargado de añadir contenido al plan de visionado no lo asignará si sus predecesores no se encuentran ya en el plan.

Parámetros	`Contenido` *s*, `Contenido` *s1*
Precondición	*predecesor(s, s1)* $\wedge$ *visionado(s1)* $\wedge$ $\neg$*visto(s1)*
Efecto	*visionado(s)*

Table 1: Descripción del operador *añadirPredecesores*

2.5.2 Añadir paralelos

Este operador es extremadamente similar al de añadir predecesores. Tiene el mismo efecto, pero esta vez lo hace para un contenido paralelo a otro. Una vez más, el segundo contenido debe estar marcado para añadir al plan, pero sin haberse añadido aún. Su utilidad es la misma: poder insertar el contenido paralelo según las restricciones del problema (de las que se ocupará el operador *ver*) para poder seguir avanzando en su resolución.

Parámetros	`Contenido` *s*, `Contenido` *s1*
Precondición	*paralelo(s, s1)* $\wedge$ *visionado(s1)* $\wedge$ $\neg$*visto(s1)*
Efecto	*visionado(s)*

Table 2: Descripción del operador *añadirParalelos*

2.5.3 Ver

El operador de ver es el que se encarga de añadir un contenido al plan de visionado en un día concreto. Recibe como parámetros un contenido *s1* y un día *d*, y tiene como precondición que el contenido *s1* esté marcado para visionado y no se haya añadido aún al plan, y que el número de contenidos del día *d* sea menor que 3 (en la extensión 3) o que el número de minutos libres del dia *d* sea mayor que la duración en minutos del contenido *s1*)en la extensión 4).

En cuanto a su efecto, tras ejecutar este operador los siguientes predicados pasan a ser ciertos: *vistoDia(s1,d)* y *visto(s1)*. Además, incrementa en 1 el número de contenidos del día *d* (en la extensión 3) o decrementa el número de minutos libres del día *d* según el número de minutos que dura el contenido *s1* (en la extensión 4). Sin embargo, este efecto no se lleva a cabo si el contenido *s1* tiene algún contenido predecesor o paralelo que aún no se haya añadido al plan, o si tiene algún contenido predecesor o paralelo añadido en el plan, pero para los cuales el día *d* no cumple con las restricciones impuestas por el problema.

Parámetros	`Contenido` *s1*, `Dia` *d*	
Precondición	$visionado(s1) \land \neg visto(s1) \land (contenidosDia(d) < 3)$	Extensión 3
	$visionado(s1) \land \neg visto(s1) \land (minLibresDia(d) > minContenido(s1))$	Extensión 4
Efecto	Si $\neg\exists$ (Contenido *s2*, Dia *d1*) tq $(predecesor(s2,s1) \land (\neg visto(s2) \lor (vistoDia(s2,d1) \land \neg anterior(d1,d)))) \lor$ $((paralelo(s2,s1) \lor paralelo(s1,s2)) \land (\neg visionado(s2)$ $\lor (vistoDia(s2,d1) \land \neg antes(d1,d) \land \neg vistoDia(s2,d)))$	
	$vistoDia(s1,d) \land visto(s1) \land increase(contenidosDia\ d,\ 1)$	Extensión 3
	$vistoDia(s1,d) \land visto(s1) \land decrease(minLibresDia(d),\ minContenido(s1))$	Extensión 4

Table 3: Descripción del operador *Ver*

3 Juegos de prueba

3.1 Prueba 1

Este juego de prueba es para la expansión 3 del proyecto.

3.1.1 Introducción

Esta prueba representa el caso de una pareja que vive sola. Durante la cuarentena, están encerrados completamente y trabajan ambos desde casa. Los dos son fanáticos del universo cinematográfico de *Marvel* y han decidido que, ya que no estrenarán la nueva película de *Black Widow* en mayo, aprovecharán el tiempo en casa para hacer un maratón de las sagas principales de *Marvel*. Todo esto para poder ver *Avengers: Endgame* la cual no habían visto porque querían esperar a verla con alguien especial... y también querían verla en blu-ray. Quieren ver las sagas de *Iron Man*, *Captain America*, *Thor* y *Avengers*, dejando de lado *Spider-Man* ya que (cuidado *spoiler*) *Iron-Man*, su personaje favorito esta muerto en *Spider-Man: Far from home* y les da demasiada pena. Como ambos son extremadamente frikis del universo *Marvel*, deciden que no verán dos películas que se precedan una a la otra en el mismo día de este modo tendrán tiempo para analizarla por la noche antes de ver la siguiente. Tampoco quieren agobiarse ni quemar las películas así que han decidido que como máximo verán 3 películas cada día. Esta prueba tiene una gran variedad de contenido con muchas películas precedentes y muchas películas paralelas. Teniendo en cuenta la larga duración de la cuarentena, en principio deberían tener el tiempo suficiente para verlas todas cumpliendo con los requisitos, por lo que se espera que el programa encuentre una solución.

Tipo de contenido	**Título**	**Acrónimo**
Película	Captain America: The First Avenger	CA
Película	Iron Man	IM
Película	Iron Man 2	IM2
Película	Thor	T
Película	The Avengers	A
Pelicula	Iron Man 3	IM3
Pelicula	Thor: The dark world	T2
Pelicula	Captain America: The winter soldier	CA2
Pelicula	Avengers: Age of Ultron	A2
Pelicula	Captain America: Civil War	CA3
Pelicula	Thor: Ragnarok	T3
Pelicula	Avengers: Infinity War	A3
Pelicula	Avengers: Endgame	A4

Table 4: Acrónimos del contenido

3.1.2 Entrada FF

```
1
2 (define (problem redflix-nivel-basico)
3    (:domain redflix)
4    (:objects CA IM IM2 T A IM3 T2 CA2 A2 CA3 T3 A3 A4 - contenido
5              d0 d1 d2 d3 d4 d5 d6 d7 - dia)
6
```

```
7
8     (:init
9
10 ;;   Relaciones de peliculas/series paralelas
11       (paralelo CA IM)
12       (paralelo CA T)
13       (paralelo CA IM2)
14       (paralelo CA2 IM3)
15       (paralelo CA2 T2)
16       (paralelo CA3 T3)
17
18
19 ;;   Relaciones de peliculas/series predecesoras
20       (predecesor CA CA2)
21       (predecesor CA CA3)
22       (predecesor CA A)
23       (predecesor CA A2)
24       (predecesor CA A3)
25       (predecesor CA A4)
26       (predecesor CA2 CA3)
27       (predecesor CA2 A2)
28       (predecesor CA2 A3)
29       (predecesor CA2 A4)
30       (predecesor CA3 A3)
31       (predecesor CA3 A4)
32       (predecesor IM IM2)
33       (predecesor IM IM3)
34       (predecesor IM A)
35       (predecesor IM A2)
36       (predecesor IM A3)
37       (predecesor IM A4)
38       (predecesor IM2 IM3)
39       (predecesor IM2 A)
40       (predecesor IM2 A2)
41       (predecesor IM2 A3)
42       (predecesor IM2 A4)
43       (predecesor IM3 A2)
44       (predecesor IM3 A3)
45       (predecesor IM3 A4)
46       (predecesor T T2)
47       (predecesor T T3)
48       (predecesor T A)
49       (predecesor T A2)
50       (predecesor T A3)
51       (predecesor T A4)
52       (predecesor T2 T3)
53       (predecesor T2 A2)
54       (predecesor T2 A3)
55       (predecesor T2 A4)
56       (predecesor T3 A3)
57       (predecesor T3 A4)
58       (predecesor A IM3)
59       (predecesor A CA2)
60       (predecesor A T2)
61       (predecesor A CA3)
62       (predecesor A T3)
63       (predecesor A A2)
64       (predecesor A A3)
```

```
     (predecesor A A4)
     (predecesor A2 CA3)
     (predecesor A2 T3)
     (predecesor A2 A3)
     (predecesor A2 A4)
     (predecesor A3 A4)



;;    Orden de los dias
      (antes d0 d1)
      (antes d1 d2)
      (antes d2 d3)
      (antes d3 d4)
      (antes d4 d5)
      (antes d5 d6)
      (antes d6 d7)


;;    Orden de los dias
      (anterior d0 d1)
      (anterior d0 d2)
      (anterior d0 d3)
      (anterior d0 d4)
      (anterior d0 d5)
      (anterior d0 d6)
      (anterior d0 d7)
      (anterior d1 d2)
      (anterior d1 d3)
      (anterior d1 d4)
      (anterior d1 d5)
      (anterior d1 d6)
      (anterior d1 d7)
      (anterior d2 d3)
      (anterior d2 d4)
      (anterior d2 d5)
      (anterior d2 d6)
      (anterior d2 d7)
      (anterior d3 d4)
      (anterior d3 d5)
      (anterior d3 d6)
      (anterior d3 d7)
      (anterior d4 d5)
      (anterior d4 d6)
      (anterior d4 d7)
      (anterior d5 d6)
      (anterior d5 d7)
      (anterior d6 d7)

;;    Peliculas o series ya vistas



;;    Peliculas/Series que se quieren ver
;;    (visionado s4)
      (visionado A4)

      (= (contenidosDia d0) 0)
```

```
      (= (contenidosDia d1) 0)
      (= (contenidosDia d2) 0)
      (= (contenidosDia d3) 0)
      (= (contenidosDia d4) 0)
      (= (contenidosDia d5) 0)
      (= (contenidosDia d6) 0)
      (= (contenidosDia d7) 0)
   )


   (:goal (not (exists (?s ?s1 - contenido) (and (visionado ?s) (not (visto ?s))) ) ) )
)
```

3.1.3 Salida FF

```
ff: parsing domain file
domain 'REDFLIX' defined
 ... done.
ff: parsing problem file
problem 'REDFLIX-NIVEL-BASICO' defined
 ... done.


no metric specified. plan length assumed.

task contains conditional effects. turning off state domination.



checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

advancing to distance:   26
                         25
                         24
                         23
                         22
                         21
                         20
                         19
                         18
                         17
                         16
                         15
                         14
                         13
                         12
                         11
                         10
                          9
                          8
                          7
                          6
                          5
                          4
                          3
```

```
                            2
                            1
                            0

ff: found legal plan as follows

step    0: ANADIRPREDECESORES A3 A4
        1: ANADIRPREDECESORES A2 A4
        2: ANADIRPREDECESORES A A4
        3: ANADIRPREDECESORES T3 A4
        4: ANADIRPREDECESORES T2 A4
        5: ANADIRPREDECESORES T A4
        6: ANADIRPREDECESORES IM3 A4
        7: ANADIRPREDECESORES IM2 A4
        8: ANADIRPREDECESORES IM A4
        9: ANADIRPREDECESORES CA3 A4
       10: ANADIRPREDECESORES CA2 A4
       11: ANADIRPREDECESORES CA A4
       12: VER CA D0
       13: VER IM D0
       14: VER IM2 D1
       15: VER T D1
       16: VER A D2
       17: VER IM3 D3
       18: VER T2 D3
       19: VER CA2 D3
       20: VER A2 D4
       21: VER CA3 D5
       22: VER T3 D5
       23: VER A3 D6
       24: VER A4 D7
       25: REACH-GOAL


time spent:    0.55 seconds instantiating 57 easy, 104 hard action templates
               0.00 seconds reachability analysis, yielding 259 facts and 4257 actions
               0.42 seconds creating final representation with 259 relevant facts, 16 relevant
    fluents
               0.03 seconds computing LNF
               0.11 seconds building connectivity graph
               0.43 seconds searching, evaluating 162 states, to a max depth of 0
               1.54 seconds total time
```

3.1.4 Resultado

En este caso encuentra una solución posible, aunque tarda 1.54 segundos. El Metric FF consigue encontrar la solución utilizando Best First Search sin probar con el Hill Climbing. La solución encontrada cumple con las restricciones impuestas por los precedentes, los paralelos y el numero de contenidos máximos por día. También se puede ver que se han distribuido los contenidos de los dos primeros días, dónde se podrían haber visto 3 películas seguidas el primer día y una el segundo, sin embargo se ven 2 por día. Gracias a Redflix, la pareja podrá hacer el maratón y finalmente ver juntos y en blu-ray la película de Endgame.

Día	Contenidos	Contenidos vistos
0	CA, IM	2
1	IM2, T	2
2	A	1
3	IM3, T2, CA2	3
4	A2	1
5	CA3, T3	2
6	A3	1
7	A4	1

Table 5: Resultado de la planificación

3.2 Prueba 2

Este juego de prueba es para la expansión 3 del proyecto.

3.2.1 Introducción

Esta prueba representa a un vendedor de un estanco en un pequeño pueblo de Lérida, ya que no suele pasar demasiada gente durante el día, y los que se pasan ya lo conocen, se ha comprado un televisor y lo ha instalado en el estanco. Un amigo suyo, dueño del bar de la estación de tren le ha recomendado una serie y le ha dicho que quizás no le gusta demasiado, pero que el último episodio es de lo mejor que ha visto. El vendedor del estanco se ha decidido por ver la serie para poder ver el último episodio, pero resulta que la serie tiene capítulos cruzados con otras películas y series, por lo que ha decidido que también verá esos episodios o películas que tengan relevancia. Ya que ha quedado el fin de semana para tomar un café con su amigo el dueño del bar, quiere poder ver el último capítulo antes de que llegue el fin de semana por lo que solo tiene los 5 días laborales. Esta prueba tiene varios contenidos que tienen tanto predecesores como contenidos paralelos por lo que debe haber pocas maneras de organizar el plan adecuadamente para que se cumplan las precedencias.

3.2.2 Entrada FF

```
;; para ejecutar : ./ff -O -o redflix.pddl -f redflix02-3.pddl
(define (problem redflix-extension3)
   (:domain redflix)
   (:objects S0 S1 S2 S3 S4 S5 - contenido
             d0 d1 d2 d3 d4 - dia)


   (:init
;;   Relaciones de peliculas/series predecesoras
      (predecesor S0 S4)
      (predecesor S1 S4)
      (predecesor S3 S4)


;;   Relaciones de peliculas/series paralelas
      (paralelo S3 S2)
      (paralelo S3 S1)
      (paralelo S4 S5)
      (paralelo S0 S2)

```

```
21
22 ;;    Orden de los dias
23       (anterior d0 d1)
24       (anterior d0 d2)
25       (anterior d0 d3)
26       (anterior d0 d4)
27       (anterior d1 d2)
28       (anterior d1 d3)
29       (anterior d1 d4)
30       (anterior d2 d3)
31       (anterior d2 d4)
32       (anterior d3 d4)
33
34
35 ;;    Orden de los dias
36       (antes d0 d1)
37       (antes d1 d2)
38       (antes d2 d3)
39       (antes d3 d4)
40
41 ;;    Peliculas o series ya vistas
42
43
44
45 ;;    Peliculas/Series que se quieren ver
46 ;;    (visionado s4)
47       (visionado S4)
48
49       (= (contenidosDia d0) 0)
50       (= (contenidosDia d1) 0)
51       (= (contenidosDia d2) 0)
52       (= (contenidosDia d3) 0)
53       (= (contenidosDia d4) 0)
54    )
55
56
57    (:goal (not (exists (?s - contenido) (and (visionado ?s) (not (visto ?s))) ) ) )
58 )
```

3.2.3 Salida FF

```
1 ff: parsing domain file
2 domain 'REDFLIX' defined
3  ... done.
4 ff: parsing problem file
5 problem 'REDFLIX-EXTENSION3' defined
6  ... done.
7
8
9 no metric specified. plan length assumed.
10
11 task contains conditional effects. turning off state domination.
12
13
14
15 checking for cyclic := effects --- OK.
16
17 ff: search configuration is EHC, if that fails then  best-first on 1*g(s) + 5*h(s) where
```

```
    metric is  plan length

Cueing down from goal distance:   10 into depth [1]
                                   9            [1]
                                   8            [1]
                                   7            [1]
                                   6            [1][2]
                                   5            [1]
                                   4            [1][2]
                                   3            [1] --- pruning stopped --- [1]

Enforced Hill-climbing failed !
switching to Best-first Search now.


advancing to distance:   10
                          9
                          8
                          7
                          6
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIRPREDECESORES S3 S4
        1: ANADIRPREDECESORES S1 S4
        2: VER S1 D0
        3: ANADIRPREDECESORES S0 S4
        4: ANADIRPARALELOS2 S3 S2
        5: VER S0 D0
        6: VER S2 D0
        7: VER S3 D1
        8: ANADIRPARALELOS2 S4 S5
        9: VER S5 D1
       10: VER S4 D2
       11: REACH-GOAL


time spent:    0.03 seconds instantiating 11 easy, 30 hard action templates
               0.00 seconds reachability analysis, yielding 83 facts and 73 actions
               0.00 seconds creating final representation with 83 relevant facts, 10 relevant
    fluents
               0.02 seconds computing LNF
               0.08 seconds building connectivity graph
               0.02 seconds searching, evaluating 75 states, to a max depth of 2
               0.15 seconds total time
```

3.2.4 Resultado

En este caso encuentra una solución posible, tarda 0.15 segundos. El Metric FF consigue encontrar la solución utilizando Best First Search después de probar con el Hill Climbing y no conseguir encontrarla. La solución encontrada cumple con las restricciones impuestas por los precedentes, los paralelos y el numero de contenidos

máximos por día.
Después de una larga semana de trabajo y maratones, el dueño del estanco ha conseguido ponerse al día para ver el último episodio de la serie que tanto le había gustado a su amigo, resulta que no era para tanto.

Día	**Contenidos**	**Contenidos vistos**
0	S0, S1, S2	3
1	S3, S5	2
2	S4	1

Table 6: Resultado de la planificación

3.3 Prueba 3

Este juego de prueba es para la expansión 4 del proyecto.

3.3.1 Introducción

Esta prueba representa el caso de un grupo de amigos que trabaja a jornada completa y están muy emocionados por la salida del videojuego *Cyberpunk 2077*. Ellos son super fans de Keanu Reeves así que deciden rememorar sus mejores películas y ponerse al día sobre él y su trabajo viendo entrevistas de varios *shows* distintos. Su objetivo es mirar la trilogía de *The Matrix*, la trilogía de *John Wick*, la película *El abogado del diablo*, 2 entrevistas en *Tonight Show*, una entrevista de XBOX sobre su participación en el videojuego y una entrevista sobre sus futuros proyectos. Pretenden verse entre semana después de trabajar para ponerse al día y el fin de semana jugar sin parar, por tanto tienen 5 días con un tiempo limitado de 200 minutos por día.

Tipo de contenido	**Título**	**Tiempo (min)**	**Acrónimo**
Película	The Matrix 1	75	M1
Película	The Matrix 2	75	M2
Película	The Matrix 3	100	M3
Película	John Wick 1	120	JW1
Película	John Wick 2	100	JW2
Película	John Wick 3	100	JW3
Película	El abogado del diablo	90	AD
Show	Tonight Show 1	25	TW1
Show	Tonight Show 2	25	TW2
Entrevista	Keanu Reeves: Cyberpunk 2077	15	XBOX
Entrevista	Keanu Reeves: Future	15	F

Table 7: Tiempo de duración del contenido

Es una prueba de contenido muy variado respecto a los tiempos de duración y se reparte a lo largo de varios días. También las películas tienen un claro orden de visualización con predecesores marcados, en cambio las entrevistas no tienen relación entre ellas, menos las dos de *Tonight Show*. Como varias de las películas ocupan la gran parte del tiempo la dificultad radica en lograr usar los huecos restantes para visualizar el resto del contenido sin desordenarlo. Ya que técnicamente, por la suma de minutos, es posible se espera encontrar una solución.

3.3.2 Entrada FF

```
(define (problem redflix-nivel-basico)
   (:domain redflix)
   (:objects s0 s1 s2 t0 t1 t2 r0 r1 w0 y0 z0 - contenido
             d0 d1 d2 d3 d4 - dia)


   (:init

;;    Orden de los dias
      (antes d0 d1)
      (antes d1 d2)
      (antes d2 d3)
      (antes d3 d4)


;;    Orden de los dias
      (anterior d0 d1)
      (anterior d0 d2)
      (anterior d0 d3)
      (anterior d0 d4)
      (anterior d1 d2)
      (anterior d1 d3)
      (anterior d1 d4)
      (anterior d2 d3)
      (anterior d2 d4)
      (anterior d3 d4)

;;    Peliculas o series ya vistas


;;    Relaciones de peliculas/series predecesoras
      (predecesor s0 s1)
      (predecesor s0 s2)
      (predecesor s1 s2)
      (predecesor t0 t1)
      (predecesor t0 t2)
      (predecesor t1 t2)
      (predecesor r0 r1)

;;    Relaciones de peliculas/series paralelas

;;    Peliculas/Series que se quieren ver
      (visionado s2)
      (visionado t2)
      (visionado r1)
      (visionado w0)
      (visionado y0)
      (visionado z0)

;;    Tiempo duracion contenido
      (= (minContenido s0) 120)
      (= (minContenido s1) 100)
      (= (minContenido s2) 100)
      (= (minContenido t0) 115)
      (= (minContenido t1)  95)
      (= (minContenido t2) 100)
```

```
      (= (minContenido r0)  25)
      (= (minContenido r1)  25)
      (= (minContenido w0)  15)
      (= (minContenido y0)  90)
      (= (minContenido z0)  15)

;;    Tiempo limite de cada dia
      (= (minLibresDia d0) 200)
      (= (minLibresDia d1) 200)
      (= (minLibresDia d2) 200)
      (= (minLibresDia d3) 200)
      (= (minLibresDia d4) 200)
   )


 (:goal (not (exists (?s ?s1 - contenido) (and (visionado ?s) (not (visto ?s))) ) ) )
)
```

3.3.3 Salida FF

```

ff: parsing domain file
domain 'REDFLIX' defined
 ... done.
ff: parsing problem file
problem 'REDFLIX-NIVEL-BASICO' defined
 ... done.


no metric specified. plan length assumed.

task contains conditional effects. turning off state domination.



checking for cyclic := effects --- OK.

ff: search configuration is EHC, if that fails then  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

Cueing down from goal distance:   17 into depth [1]
                                  16            [1]
                                  15            [1]
                                  14            [1]
                                  13            [1]
                                  12            [1]
                                  11            [1]
                                  10            [1]
                                   9            [1]
                                   8            [1]
                                   7            [1]
                                   6            [1]
                                   5            [1] --- pruning stopped --- [1]

Enforced Hill-climbing failed !
switching to Best-first Search now.


```

```
advancing to distance:   17
                         16
                         15
                         14
                         13
                         12
                         11
                         10
                          9
                          8
                          7
                          6
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: VER W0 D0
        1: VER Z0 D0
        2: VER Y0 D1
        3: ANADIRPREDECESORES R0 R1
        4: VER R0 D0
        5: VER R1 D2
        6: ANADIRPREDECESORES T1 T2
        7: ANADIRPREDECESORES T0 T2
        8: VER T0 D2
        9: VER T1 D3
       10: VER T2 D4
       11: ANADIRPREDECESORES S1 S2
       12: ANADIRPREDECESORES S0 S2
       13: VER S0 D0
       14: VER S1 D1
       15: VER S2 D3
       16: REACH-GOAL


time spent:    0.00 seconds instantiating 7 easy, 55 hard action templates
               0.00 seconds reachability analysis, yielding 148 facts and 94 actions
               0.00 seconds creating final representation with 143 relevant facts, 5 relevant
    fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.51 seconds searching, evaluating 29681 states, to a max depth of 1
               0.51 seconds total time
```

3.3.4 Resultado

En este caso encuentra una solución posible, aunque tarda 0,51 segundos. De hecho debido a la complejidad que se propone el Hill Climing falla en encontrar una solución, así que el Metric FF pasa a aplicar Best First Search. La solución hallada respeta todos los precedentes y no sobrepasa el límite de tiempo impuesto. El grupo de amigos consigue ver la trilogía de *The Matrix* los días 2,3 y 4, *John Wick* los días 1, 2 y 3, el *Tonight Show* los días 0 y 1, respetando todas las precedencias. Además, nuestro programa intenta llenar antes los días que tienen todavía

espacio libre, por ello las entrevistas y la otra película se visualizan el primer día, en vez de repartirse por lo demás huecos ya que cabrían en otros momentos.

Día	Contenidos	Tiempos (min)	Tiempo restante (min)
0	XBOX, AD, F, TS1	$15 + 90 + 15 + 25$	55
1	JW1, TS2	$120 + 25$	55
2	M1, JW2	$75 + 100$	25
3	M2, JW3	$75 + 100$	25
4	M3	100	100

Table 8: Resultado de la planificación

3.4 Prueba 4

Este juego de prueba es para la expansión 4 del proyecto.

3.4.1 Introducción

Esta prueba representa el caso de un estudiante que quiere aprovechar un fin de semana largo para ver una serie y contenido relacionado con esta, pero sin dejar sus estudios de lado. Tendremos una serie de 3 episodios. Cada episodio es distinto, ya que el hilo conductor de la serie es su temática, como ocurre en *Black Mirror*. Por tanto el primer episodio dura 170 min, el segundo 100 y el tercero 120 minutos. Además queremos ver una película relacionada, es un *spin-off* del primer episodio, así que tienen que verse en paralelo. La película dura 2 horas (120 min). Tenemos 3 días, es decir un puente, pero cada día solo tenemos disponibles 200 min para ver contenido ya que el resto debe estudiar.

Debido al poco tiempo para ver el contenido, ya que solo tiene 200 min cada día durante 3 días, no tendrá tiempo. Debido que la película paralela ocupa casi todo un día (170 min) y no se puede compaginar con otro episodio, porque ninguno dura menos de 30 minutos.

3.4.2 Entrada FF

```
1
2  (define (problem redflix-nivel-basico)
3     (:domain redflix)
4     (:objects s0 s1 s2 t0 - contenido
5               d0 d1 d2 - dia)
6
7
8     (:init
9
10 ;;    Orden de los dias
11       (antes d0 d1)
12       (antes d1 d2)
13
14
15 ;;    Orden de los dias
16       (anterior d0 d1)
17       (anterior d0 d2)
18       (anterior d1 d2)
```

```
;;    Peliculas o series ya vistas


;;    Relaciones de peliculas/series predecesoras
      (predecesor s0 s1)
      (predecesor s0 s2)
      (predecesor s1 s2)

;;    Relaciones de peliculas/series paralelas

;;    Peliculas/Series que se quieren ver
      (visionado s2)
      (visionado t0)

;;    Tiempo duracion contenido
      (= (minContenido s0) 170)
      (= (minContenido s1) 100)
      (= (minContenido s2) 120)
      (= (minContenido t0) 120)

;;    Tiempo limite de cada dia
      (= (minLibresDia d0) 200)
      (= (minLibresDia d1) 200)
      (= (minLibresDia d2) 200)
   )


 (:goal (not (exists (?s ?s1 - contenido) (and (visionado ?s) (not (visto ?s))) ) ) )
)
```

3.4.3 Salida FF

```
ff: parsing domain file
domain 'REDFLIX' defined
 ... done.
ff: parsing problem file
problem 'REDFLIX-NIVEL-BASICO' defined
 ... done.


no metric specified. plan length assumed.

task contains conditional effects. turning off state domination.



checking for cyclic := effects --- OK.

ff: search configuration is EHC, if that fails then  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

Cueing down from goal distance:    7 into depth [1]
                                   6            [1]
                                   5            [1]
                                   4            [1]
                                   3            [1] --- pruning stopped --- [1]
```

```
26
27 Enforced Hill-climbing failed !
28 switching to Best-first Search now.
29
30
31 advancing to distance:    7
32                           6
33                           5
34                           4
35                           3
36
37 best first search space empty! problem proven unsolvable.
38
39
40
41 time spent:    0.00 seconds instantiating 3 easy, 12 hard action templates
42                0.00 seconds reachability analysis, yielding 38 facts and 19 actions
43                0.00 seconds creating final representation with 37 relevant facts, 3 relevant
       fluents
44                0.00 seconds computing LNF
45                0.00 seconds building connectivity graph
46                0.00 seconds searching, evaluating 55 states, to a max depth of 1
47                0.00 seconds total time
```

3.4.4 Resultado

Como podemos observar en la salida, el resultado es `unsolvable` que es justo nuestro resultado esperado. El Metrics FF prueba varias maneras de resolver el problema, primero Hill-climbing y cuando falla pasa a Best-first Search. Al no encontrar ninguna posibilidad en que se satisfagan todas las condiciones declara el problema irresoluble. Para el estudiante queda demostrado que es imposible ver todas las series que quiere y estudiar la cantidad de horas que se ha propuesto, por tanto se verá obligado a escoger.

3.5 Generador de Juegos de Prueba

3.5.1 Introducción

Para generar juegos de prueba de forma aleatoria, hemos implementado un pequeño script en Python. Este script puede generar problemas válidos tanto para los dominios de las extensiones 3 y 4. El programa se divide en dos partes: un algoritmo que genera la estructura del problema y una función que convierte las estructuras que lo representan en Python a un archivo PDDL. El código completo del programa está disponible en el Annexo.

3.5.2 Algoritmo

```
n_contenido = Número aleatorio del 8 al 14
n_dias = Número aleatorio del 3 al 7
n_visionado = Número aleatorio del 3 al 6

Por cada número de 0 a n_contenido:
    //Asignación de predecesores
    n_predecesores = número aleatorio entre 0 y 1
    si n_predecesores > 0:
        elegimos un predecesor pred aleatorio
```

```
        si pred no es predecesor o paralelo de i, i no es predecesor o paralelo de pred, y pred != i:
            pred pasa a ser predecesor de i


    //Asignación de paralelos
    n_paralelos = número aleatorio entre 0 y 1
    si n_paralelos > 0:
        elegimos un paralelo par aleatorio
        si par no es predecesor o paralelo de i, i no es predecesor o paralelo de par, y par != i:
            par pasa a ser paralelo de i

    Por cada numero de 0 a n_contenido:
        asignamos un número de minutos aleatorio entre 50 y 110 al contenido

Por cada número de 0 a n_visionado:
    elegimos un contenido aleatorio y lo añadimos a la lista de visionado
```

4 Completado de los niveles

4.1 Nivel Básico

En el nivel básico se nos pedía implementar un programa que creara un plan teniendo en cuenta que todos los contenidos pueden tener o 1 o 0 predecesores y que el plan creado incluyera todos los predecesores del contenido que se quiere ver. Para esto hemos creado una variable:

```
(:types contenido - object
      )
```

Esta variable nos sirve para representar los contenidos. También hemos creado 3 predicados:

```
(:predicates
    (visto ?peli - contenido)
    (visionado ?peli - contenido)
    (predecesor ?peli1 - contenido ?peli2 - contenido)
)
```

El predicado *visto* nos indica si un contenido se ha añadido al plan o no, el predicado *visionado* indica si un contenido se quiere ver o no y por último el predicado *predecesor* nos indica si un contenido es predecesor de otro. Por ultimo hemos creado 2 acciones:

```
(:action anadirPredecesores ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
     :parameters (?s - contenido ?s1 - contenido)
     :precondition (and (predecesor ?s ?s1) (visionado ?s1) (not (visto ?s1)))
     :effect  (visionado ?s)
     )

(:action ver
    :parameters (?s1 - contenido )
    :precondition (and (visionado ?s1) (not (visto ?s1)))
    :effect (when (not
                   (exists (?s2 - contenido )
                    (and (predecesor ?s2 ?s1) (not (visto ?s2)))
                   )
                  )
                  (and (visto ?s1))
                )
)
```

La primera acción se utiliza para marcar a los predecesores de un contenido como pendientes de ver. La segunda sirve para añadir a ver un contenido siempre y cuando sus predecesores se hayan visto.

4.2 Extensión 1

Para esta extensión se nos pedía que el programa pudiera crear un plan teniendo en cuenta que cada contenido podía tener varios predecesores, y el plan debía incluir a todos los predecesores de un contenido que se quería ver en días anteriores. Para hacerlo hemos añadido una variable:

```
día - object
```

Esta variable nos servirá para mantener el orden de los días en los que se ven los contenidos. También hemos añadido dos predicados:

```
(:predicates
   (vistoDia ?pelicula - contenido ?d - dia)
   (anterior ?dia1 - dia ?dia2 - dia)
   )
```

El predicado *vistoDia* nos guardará la información de que día se ha visto el contenido y el predicado *anterior* lo usaremos para saber que días se preceden. Por último hemos modificado la acción ver:

```
(:action ver
 :parameters (?s1 - contenido ?d - dia)
 :precondition (and (visionado ?s1) (not (visto ?s1)))
 :effect (when (not
                (exists (?s2 - contenido ?d1 - dia)
                 (and (predecesor ?s2 ?s1)
                   (or (not (visto ?s2)) (and (vistoDia ?s2 ?d1) (not (anterior ?d1 ?d))))
                 )
                )
               )
               (and (vistoDia ?s1 ?d) (visto ?s1))
             )
 )
```

En esta acción hemos añadido hecho que antes de añadir un contenido a ver, se compruebe que no existe un predecesor que se haya visto un día posterior al que se está tratando.

4.3 Extensión 2

Para esta extensión se nos pedía que a partir de la extensión 1 tuviésemos también en cuenta la posibilidad de que los contenidos puedan tener uno o varios contenidos paralelos, y que estos se debían ver un día antes del contenido en cuestión o el mismo día. Para eso hemos añadido dos predicados:

```
(:predicates
   (paralelo ?serie1 - contenido ?serie2 - contenido)
   (antes ?dia1 - dia ?dia2 - dia)
   )
```

El predicado *paralelo* indica que contenidos son paralelos entre sí y el predicado *antes* lo usaremos para saber el orden de los días, ya que con anterior solo podíamos saber si un día era anterior a otro pero no podíamos saber que día era el previo a otro. Hemos añadido también dos acciones:

```
(:action anadirParalelos ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
     :parameters (?s - contenido ?s1 - contenido)
     :precondition (and (paralelo ?s ?s1) (visionado ?s1)(not (visto ?s1)))
     :effect  (visionado ?s)
     )
```

```
(:action anadirParalelos ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
     :parameters (?s - contenido ?s1 - contenido)
     :precondition (and (paralelo ?s ?s1) (visionado ?s)(not (visto ?s)))
     :effect  (visionado ?s1)
     )
```

Estas acciones las hemos usado para indicar que cuando un contenido hay que verlo y existe otro contenido paralelo a este, también hay que verlo. Por ultimo hemos modificado la acción de ver:

```
(:action ver
     :parameters (?s1 - contenido ?d - dia)
     :precondition (and (visionado ?s1) (not (visto ?s1)))
     :effect (when (not
                     (exists (?s2 - contenido ?d1 - dia)
                     (or
                       (and (predecesor ?s2 ?s1)
                        (or (not (visto ?s2)) (and (vistoDia ?s2 ?d1) (not (anterior ?d1 ?d))))
                       )
                       (and (or (paralelo ?s2 ?s1)
                        (paralelo ?s1 ?s2))
                        (or (not (visionado ?s2))
                            (and (vistoDia ?s2 ?d1) (not (antes ?d1 ?d)) (not (vistoDia ?s2 ?d)))
                        )
                       )
                      )
                     )
                   )
                   (and (vistoDia ?s1 ?d) (visto ?s1))
                )
     )
```

Hemos añadido la condición de que si existe un contenido paralelo al que se esta tratando que se no esta marcado como pendiente de ver o se ha visto un día distinto al día antes o al mismo día que se esta tratando, no se puede añadir a ver.

4.4 Extensión 3

Para esta extensión se nos pedía que partiendo de la extensión 2 tuviesemos en cuenta que solo se pueden ver 3 contenidos como máximo cada día. Para ello hemos añadido una función:

```
(:functions
    (contenidosDia ?d - dia)
)
```

Usaremos esta función para contar cuantos contenidos añadimos a cada día y comprobar antes de añadir un contenido que no haya más de 2 contenidos planeados para el día que se esté tratando. Para ello hemos modificado la acción ver de la siguiente manera:

```
(:action ver
     :parameters (?s1 - contenido ?d - dia)
     :precondition (and (visionado ?s1) (not (visto ?s1)) (< (contenidosDia ?d) 3))
     :effect (when (not
                     (exists (?s2 - contenido ?d1 - dia)
                     (or
                       (and (predecesor ?s2 ?s1)
                        (or (not (visto ?s2)) (and (vistoDia ?s2 ?d1) (not (anterior ?d1 ?d))))
                       )
                       (and (or (paralelo ?s2 ?s1)
                        (paralelo ?s1 ?s2))
                        (or (not (visionado ?s2))
                            (and (vistoDia ?s2 ?d1) (not (antes ?d1 ?d)) (not (vistoDia ?s2 ?d)))
                        )
                       )
                     )
                   )
                   (and (vistoDia ?s1 ?d) (visto ?s1) (increase (contenidosDia ?d) 1))
                 )
     )
```

4.5 Extensión 4

Para esta ampliación del código se nos pedía que a partir de la extensión 2 añadiésemos al contenido un tiempo de duración en minutos y limitásemos a 200 minutos el tiempo posible de visualización de cada día. Por ello fue necesario añadir dos funciones:

```
(:functions
    (minLibresDia ?d - dia)
    (minContenido ?peli - contenido)
)
```

De manera que pudiésemos asignar a cada contenido su duración y archivar el tiempo disponible de un día.

Respecto a las `action` hubo que cambiar una precondición y uno de los efectos. En la `action ver` se añadió en la precondición que el contenido que se fuese a ver debía durar menos que la cantidad de tiempo libre que quedaba en ese día, de manera que no se pudiese asignar contenido que sobrepasara el límite estipulado.
`:precondition (and (visionado ?s1) (not (visto ?s1)) (> (minLibresDia ?d) (minContenido ?s1)) )`

En los efectos de las misma se añadió una cláusula que substraía al tiempo disponible de ese día los minutos del contenido asignado para actualizar el valor del tiempo libre.

`(and (vistoDia ?s1 ?d) (visto ?s1) (decrease (minLibresDia ?d) (minContenido ?s1)))`

La última modificación fue cambiar los datos de entrada que ahora requerían de dos parámetros nuevos.

```
;;    Tiempo duracion contenido
      (= (minContenido s0) 120)
```

```
;;    Tiempo limite de cada dia
      (= (minLibresDia d0) 200)
```

5 Conclusión

Tras haber realizado el proyecto podemos concluir que nuestro trabajo tiene ciertas similitudes con la práctica de sistemas basados en el conocimiento realizada anteriormente. Principalmente encontramos un parecido con la metodología de resolución constructiva usada en los SBC, ya que también partimos de un estado inicial y mediante distintos operadores lo vamos modificando teniendo en cuenta ciertas restricciones hasta llegar al estado final. A pesar de esto también existen ciertas diferencias, ya que en la resolución constructiva los operadores se eligen con el criterio de cuál es el óptimo para llegar al estado final. Sin embargo en esta práctica se realiza un *backtracking* con *Hill Climbing* o *Best First Search* que permite encontrar la secuencia de operadores que lleva antes al estado final.

A Código del programa

A.1 Nivel básico

```
(define (domain redflix)
  (:requirements :adl :typing)
  (:types contenido - object
          )
  (:predicates
   (visto ?peli - contenido)
   (visionado ?peli - contenido)      ;; lo que el queremos visionar si o si
   (predecesor ?peli1 - contenido ?peli2 - contenido)  ;; la peli1 precede a la peli2
   )

   (:action anadirPredecesores ;; si el contenido s1 hay que verlo entonces ponemos antes al
    contenido s
     :parameters (?s - contenido ?s1 - contenido)
     :precondition (and (predecesor ?s ?s1) (visionado ?s1) (not (visto ?s1)))
     :effect  (visionado ?s)
     )

     (:action ver
       :parameters (?s1 - contenido )
       :precondition (and (visionado ?s1) (not (visto ?s1)))
        ;; No puede existir ningun predecesor que no se haya visto  o que se haya visto un dia que
    no sea anterior a dia ?d
        ;; y tampoco puede existir un paralelo que o no este marcado para ver o se haya visto un
    dia que no es el dia  de antes o el mismo dia.
       :effect (when (not
                       (exists (?s2 - contenido )
                        (and (predecesor ?s2 ?s1) (not (visto ?s2)))
                       )
                     )
                     (and (visto ?s1))
                   )
       )
)
```

A.2 Extensión 1

```
(define (domain redflix)
  (:requirements :adl :typing)
  (:types contenido - object
          dia - object
          )
  (:predicates
   (visto ?peli - contenido)
   (vistoDia ?pelicula - contenido ?d - dia)      ;; contenido que ya se ha visto
   (visionado ?peli - contenido)      ;; lo que el queremos visionar si o si
   (predecesor ?peli1 - contenido ?peli2 - contenido)  ;; la peli1 precede a la peli2
   (anterior ?dia1 - dia ?dia2 - dia) ;;nos dice que el dia1 es anterior que el dia2
   )

   (:action anadirPredecesores ;; si el contenido s1 hay que verlo entonces ponemos antes al
    contenido s
     :parameters (?s - contenido ?s1 - contenido)
     :precondition (and (predecesor ?s ?s1) (visionado ?s1) (not (visto ?s1)))
     :effect  (visionado ?s)
```

```
    )

  (:action ver
    :parameters (?s1 - contenido ?d - dia)
    :precondition (and (visionado ?s1) (not (visto ?s1)))
     ;; No puede existir ningun predecesor que no se haya visto  o que se haya visto un dia que no sea anterior a dia ?d
     ;; y tampoco puede existir un paralelo que o no este marcado para ver o se haya visto un dia que no es el dia  de antes o el mismo dia.
    :effect (when (not
                    (exists (?s2 - contenido ?d1 - dia)
                     (and (predecesor ?s2 ?s1) (or (not (visto ?s2)) (and (vistoDia ?s2 ?d1) (not (anterior ?d1 ?d)))) )
                    )
                  )
                  (and (vistoDia ?s1 ?d) (visto ?s1))
                )
    )
)
```

A.3 Extensión 2

```
(define (domain redflix)
  (:requirements :adl :typing)
  (:types contenido - object
          dia - object
          )
  (:predicates
   (visto ?peli - contenido)
   (vistoDia ?pelicula - contenido ?d - dia)      ;; contenido que ya se ha visto
   (visionado ?peli - contenido)      ;; lo que el queremos visionar si o si
   (predecesor ?peli1 - contenido ?peli2 - contenido)  ;; la peli1 precede a la peli2
   (paralelo ?serie1 - contenido ?serie2 - contenido)  ;; la peli1 precede a la peli2
   (anterior ?dia1 - dia ?dia2 - dia) ;;nos dice que el dia1 es anterior que el dia2
   (antes ?dia1 - dia ?dia2 - dia) ;;nos dice que el dia1 es el dia de antes del dia2
   )

   (:action anadirPredecesores ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
    :parameters (?s - contenido ?s1 - contenido)
    :precondition (and (predecesor ?s ?s1) (visionado ?s1) (not (visto ?s1)))
    :effect  (visionado ?s)
    )

   (:action anadirParalelos ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
    :parameters (?s - contenido ?s1 - contenido)
    :precondition (and (paralelo ?s ?s1) (visionado ?s1)(not (visto ?s1)))
    :effect  (visionado ?s)
    )

   (:action anadirParalelos2 ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
    :parameters (?s - contenido ?s1 - contenido)
    :precondition (and (paralelo ?s ?s1) (visionado ?s)(not (visto ?s)))
    :effect  (visionado ?s1)
    )

```

```
   (:action ver
     :parameters (?s1 - contenido ?d - dia)
     :precondition (and (visionado ?s1) (not (visto ?s1)))
      ;; No puede existir ningun predecesor que no se haya visto  o que se haya visto un dia que no sea anterior a dia ?d
      ;; y tampoco puede existir un paralelo que o no este marcado para ver o se haya visto un dia que no es el dia  de antes o el mismo dia.
     :effect (when (not
                     (exists (?s2 - contenido ?d1 - dia)
                     (or
                       (and (predecesor ?s2 ?s1) (or (not (visto ?s2)) (and (vistoDia ?s2 ?d1) (not (anterior ?d1 ?d))))
                       (and (or (paralelo ?s2 ?s1) (paralelo ?s1 ?s2)) (or (not (visionado ?s2))(and (vistoDia ?s2 ?d1) (not (antes ?d1 ?d)) (not (vistoDia ?s2 ?d)))))
                     )
                   )
                   (and (vistoDia ?s1 ?d) (visto ?s1))
                 )
     )
)
```

A.4 Extensión 3

```
(define (domain redflix)
  (:requirements :adl :typing)
  (:types contenido - object
          dia - object
          )
  (:functions
    (contenidosDia ?d - dia)
    )
  (:predicates
   (visto ?peli - contenido)
   (vistoDia ?pelicula - contenido ?d - dia)      ;; contenido que ya se ha visto
   (visionado ?peli - contenido)      ;; lo que el queremos visionar si o si
   (predecesor ?peli1 - contenido ?peli2 - contenido)  ;; la peli1 precede a la peli2
   (paralelo ?serie1 - contenido ?serie2 - contenido)  ;; la peli1 precede a la peli2
   (anterior ?dia1 - dia ?dia2 - dia) ;;nos dice que el dia1 es anterior que el dia2
   (antes ?dia1 - dia ?dia2 - dia) ;;nos dice que el dia1 es el dia de antes del dia2
   )

   (:action anadirPredecesores ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
     :parameters (?s - contenido ?s1 - contenido)
     :precondition (and (predecesor ?s ?s1) (visionado ?s1) (not (visto ?s1)))
     :effect  (visionado ?s)
     )

   (:action anadirParalelos ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
     :parameters (?s - contenido ?s1 - contenido)
     :precondition (and (paralelo ?s ?s1) (visionado ?s1)(not (visto ?s1)))
     :effect  (visionado ?s)
     )

   (:action anadirParalelos2 ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
```

```
    :parameters (?s - contenido ?s1 - contenido)
    :precondition (and (paralelo ?s ?s1) (visionado ?s)(not (visto ?s)))
    :effect  (visionado ?s1)
    )

  (:action ver
    :parameters (?s1 - contenido ?d - dia)
    :precondition (and (visionado ?s1) (not (visto ?s1)) (< (contenidosDia ?d) 3))
     ;; No puede existir ningun predecesor que no se haya visto  o que se haya visto un dia que no sea anterior a dia ?d
     ;; y tampoco puede existir un paralelo que o no este marcado para ver o se haya visto un dia que no es el dia  de antes o el mismo dia.
    :effect (when (not
                    (exists (?s2 - contenido ?d1 - dia)
                    (or
                      (and (predecesor ?s2 ?s1) (or (not (visto ?s2)) (and (vistoDia ?s2 ?d1) (not (anterior ?d1 ?d)))) )
                      (and (or (paralelo ?s2 ?s1) (paralelo ?s1 ?s2)) (or (not (visionado ?s2))(and (vistoDia ?s2 ?d1) (not (antes ?d1 ?d)) (not (vistoDia ?s2 ?d)))))
                    )
                   )
                  (and (vistoDia ?s1 ?d) (visto ?s1) (increase (contenidosDia ?d) 1))
                )
    )
)
```

A.5 Extensión 4

```
(define (domain redflix)
  (:requirements :adl :typing :fluents)
  (:types contenido - object
          dia - object)
  (:functions
    (minLibresDia ?d - dia)
    (minContenido ?peli - contenido)
    )
  (:predicates
   (visto ?peli - contenido)
   (vistoDia ?pelicula - contenido ?d - dia)      ;; contenido que ya se ha visto
   (visionado ?peli - contenido)      ;; lo que el queremos visionar si o si
   (predecesor ?peli1 - contenido ?peli2 - contenido)  ;; la peli1 precede a la peli2
   (paralelo ?serie1 - contenido ?serie2 - contenido)  ;; la peli1 precede a la peli2
   (antes ?dia1 - dia ?dia2 - dia) ;;nos dice que el dia1 es el dia de antes del dia2
   (anterior ?dia1 - dia ?dia2 - dia) ;;nos dice que el dia1 es anterior que el dia2
   )

   (:action anadirPredecesores ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
    :parameters (?s - contenido ?s1 - contenido)
    :precondition (and (predecesor ?s ?s1) (visionado ?s1)(not (visto ?s1)))
    :effect  (visionado ?s)
    )

   (:action anadirParalelos ;; si el contenido s1 hay que verlo entonces ponemos antes al contenido s
    :parameters (?s - contenido ?s1 - contenido)
    :precondition (and (paralelo ?s ?s1) (visionado ?s1)(not (visto ?s1)))
```

```
     :effect  (visionado ?s)
     )

     (:action ver
       :parameters (?s1 - contenido ?d - dia)
       :precondition (and (visionado ?s1) (not (visto ?s1)) (> (minLibresDia ?d) (minContenido ?s1
    )) )
        ;; No puede existir ningun predecesor que no se haya visto  o que se haya visto un dia que
     no sea anterior a dia ?d
        ;; y tampoco puede existir un paralelo que o no este marcado para ver o se haya visto un
    dia que no es el dia  de antes o el mismo dia.
       :effect (when (not
                       (exists (?s2 - contenido ?d1 - dia)
                       (or
                         (and (predecesor ?s2 ?s1) (or (not (visto ?s2)) (and (vistoDia ?s2 ?d1) (
    not (anterior ?d1 ?d)))) )
                         (and (paralelo ?s2 ?s1) (or (not (visionado ?s2))(and (vistoDia ?s2 ?d1)
    (not (antes ?d1 ?d)) (not (vistoDia ?s2 ?d)))))
                         )
                       )
                     )
                     (and (vistoDia ?s1 ?d) (visto ?s1) (decrease (minLibresDia ?d) (minContenido
    ?s1)))
                   )
       )
)
```

A.6 Generador de problemas aleatorios

```
import random

def print_hi(name):
    # Use a breakpoint in the code line below to debug your script.
    print(f'Hi, {name}')  # Press Ctrl+F8 to toggle the breakpoint.

def write_file(n_contenido, n_dias, predecesores, paralelos, min_contenido, visionado, extension):

    f = open("redflix_random.pddl", "w")

    f.write("(define (problem redflix-aleatorio)\n")
    f.write("\t(:domain redflix)\n")
    f.write("\t(:objects ")
    for i in range(n_contenido):
        f.write("s" + str(i) + " ")
    f.write("- contenido\n")
    f.write("\t")
    for i in range(n_dias):
        f.write("d" + str(i) + " ")
    f.write("- dia)\n\n")

    f.write("\t(:init\n\n")
    for i in range(n_dias - 1):
        f.write("\t\t(antes d"+str(i)+" d"+str(i+1)+")\n")
    f.write("\n")
    for i in range(n_dias):
        for j in range(i+1, n_dias):
            f.write("\t\t(anterior d"+str(i)+" d"+str(j)+")\n")
    f.write("\n")
```

```
    for i in range(n_contenido):
        curr_predecesores = predecesores.get(i, [])
        for j in curr_predecesores:
            f.write("\t\t(predecesor s"+str(i)+" s"+str(j)+")\n")
    f.write("\n")
    for i in range(n_contenido):
        curr_paralelos = paralelos.get(i, [])
        for j in curr_paralelos:
            f.write("\t\t(paralelo s"+str(i)+" s"+str(j)+")\n")
    f.write("\n")
    for i in visionado:
        f.write("\t\t(visionado s"+str(i)+")\n")
    f.write("\n")

    if extension == 3:
        for i in range(n_dias):
            f.write("\t\t(= (contenidosDia d"+str(i)+") 0)\n")

    if extension == 4:
        for i in range(n_contenido):
            f.write("\t\t(= (minContenido s"+str(i)+") "+str(min_contenido[i])+")\n")
        f.write("\n")

        for i in range(n_dias):
            f.write("\t\t(= (minLibresDia d"+str(i)+") 200)\n")

    f.write("\t)\n\n")

    f.write("\t(:goal (not (exists (?s ?s1 - contenido) (and (visionado ?s) (not (visto ?s))) ) ) )\n")

    f.write(")")



if __name__ == '__main__':
    ext = 0
    while ext != 3 and ext != 4:
        print(" Usando  la extensi n 3 o 4?")
        ext = int(input())
        if ext != 3 and ext != 4:
            print("Introduzca solo el numero, por favor.")

    n_contenido = random.randint(8,14)
    n_dias = random.randint(3, 7)
    n_visionado = random.randint(3, 6)
    predecesores = {}
    paralelos = {}
    min_contenido = []
    visionado = []
    for i in range(n_contenido):
        # Asignamos contenido predecesor
        n_predecesores = random.randint(0, 1)
        if n_predecesores > 0:
            predecesor = random.randrange(n_contenido)
            if predecesor not in predecesores.get(i, []) and i not in predecesores.get(predecesor, []) \
                    and predecesor not in paralelos.get(i, []) and i not in paralelos.get(
```

```
    predecesor, []) \
                    and i != predecesor:
                predecesores[i] = predecesores.get(i, [])
                predecesores[i].append(predecesor)

        #Asignamos contenido paralelo
        n_paralelos = random.randint(0, 1)
        for p in range(n_paralelos):
            paralelo = random.randrange(n_contenido)
            if paralelo not in paralelos.get(i, []) and paralelo not in predecesores.get(i, []) \
                    and i not in predecesores.get(paralelo, []) and i not in paralelos.get(
    paralelo, []) \
                    and i != paralelo:
                paralelos[i] = paralelos.get(i, [])
                paralelos[i].append(paralelo)

        for j in range(n_contenido):
            min_contenido.append(random.randint(50, 110))

    for i in range(n_visionado):
        curr_visionado = random.randrange(n_contenido)
        if curr_visionado not in visionado:
            visionado.append(curr_visionado)

    write_file(n_contenido, n_dias, predecesores, paralelos, min_contenido, visionado, ext)
```