Proyecto de Planificación

Facultad Informática de Barcelona (UPC)
Grado en ingeniería informática
Curso 2024-2025

Inteligencia Artificial
13, enero del 2025

Verenisse Cáceres
Lluc Santamaria Riba
Adrià Cebrián

Tabla de Contenidos

Identificación del problema

Hoy en día, el entretenimiento audiovisual es una de las formas más comunes de ocio, especialmente en un mundo donde las plataformas de streaming y las maratones de series y películas han ganado popularidad. Sin embargo, cuando se trata de planificar el visionado de contenidos específicos, como películas o capítulos de series, los usuarios suelen enfrentarse a diversas restricciones. Estas incluyen la disponibilidad limitada de tiempo, la necesidad de respetar el orden narrativo de los contenidos y la gestión de contenidos que pueden o deben verse de manera simultánea o en días consecutivos.

El problema radica en que los contenidos audiovisuales no siempre pueden ser consumidos en cualquier orden, ya que muchos están interconectados a través de predecesores que forman parte de una narrativa más amplia. Además, algunos contenidos paralelos, como los episodios de un crossover entre series o películas relacionadas, requieren ser vistos juntos o con una proximidad temporal que respete su relación de simultaneidad. Estas restricciones complican la planificación de un visionado que permita al usuario disfrutar de una experiencia coherente y satisfactoria sin desperdiciar tiempo.

Para abordar este desafío, es necesario identificar las restricciones y características que deben ser consideradas en el proceso de planificación. Por un lado, se debe tener en cuenta:

- El tiempo disponible para consumir los contenidos.
- Los contenidos que el usuario quiere ver específicamente.
- Los contenidos que el usuario ya ha visto y que no deben ser incluidos nuevamente.
- Las relaciones de predecesor, que determinan el orden en el que algunos contenidos deben ser vistos.
- Las relaciones de paralelismo, que requieren que ciertos contenidos sean agendados juntos o en días consecutivos.

Por otro lado, es fundamental conocer las propiedades de los contenidos audiovisuales, como su duración, su rol dentro de la narrativa global (objetivo o predecesor), y su relación con otros contenidos en términos de paralelismo o dependencia. Esta información permite construir un modelo que garantice la coherencia narrativa y una experiencia satisfactoria para el usuario.

El sistema resultante deberá ser capaz de generar un plan de visionado que respete todas estas restricciones y que optimice el tiempo del usuario, asegurando que todos los contenidos objetivo sean consumidos de forma coherente. Así, la solución debe contemplar tanto las necesidades específicas del usuario como las características de los contenidos, logrando una planificación eficiente y personalizada.

Desarrollo de modelos

Para realizar esta práctica, hemos decidido utilizar el desarrollo incremental basado en prototipado rápido. Tal y como describe el enunciado de la práctica, hemos empezado por el nivel básico y hemos ido haciendo iteraciones para cada extensión. Al ser tres integrantes en el grupo, hemos dividido el trabajo en tres tareas por iteración:

- Modelado del dominio: El dominio recoge las variables, predicados y acciones en un fichero domain.pddl

- Modelado de un problema real a resolver: El problema recoge los objetos, el estado inicial y el estado final en un fichero problem.pddl. Permite verificar el correcto funcionamiento del dominio en casos comunes.

- Generador de juegos de prueba: Un programa que permite generar juegos de prueba aleatorios. Permite detectar errores que pasaron desapercibidos con el problema de caso real.

En cada iteración, cada miembro hace su parte del trabajo respectivo. Por esto, para cada extensión hay un modelado de dominio, un modelado de problema real, un generador de juegos de prueba y varios juegos de prueba generados por el generador como ejemplo. Efectivamente, cada iteración parte de la previa reutilizando, ajustando y ampliando las características. El trabajo asignado por iteración es rotativo. De esta forma todos los miembros del equipo aprendemos y trabajamos en los diferentes requisitos de la práctica.

La siguiente documentación de la práctica se ha dividido en las diferentes iteraciones realizadas. Para cada iteración, se explica la forma en la que se ha modelado el dominio, la forma en la que se modelan los problemas a resolver y el conjunto de problemas de prueba.

Nivel básico

Esta es la primera iteración de la práctica. Parte desde cero. Se nos pide encontrar un plan de visionado donde los contenidos tienen como mucho un predecesor y no tienen contenidos paralelos. El plan debe permitir cubrir los contenidos objetivos en un encadenamiento de contenidos.

Modelado del dominio

Para modelar correctamente el dominio, hemos utilizado un total de dos variables, ocho predicados y dos operadores.

Variables

Hemos decidido utilizar las siguientes variables:

- **contenido**: Permite representar el conocimiento de los contenidos audiovisuales disponibles en la plataforma Redflix.

- **días**: Permite representar el conocimiento de los días los cuales el usuario quiere agendar

contenidos a visualizar.

Predicados

Una vez ya determinadas las variables, hemos decidido utilizar los siguientes predicados para representar el conocimiento y el estado del problema:

- **haVisto(contenido)**: Permite representar el conocimiento de cuáles contenidos el usuario ya ha visto. Un contenido evalúa a cierto si y sólo si el usuario ya ha visto el contenido. Este predicado es necesario para evitar añadir contenidos al plan que el usuario ya ha visto. En el fichero de problema se inicializa el predicado en ciertas variables de tipo contenido, y durante la ejecución no se modifican, ya que es un predicado estático.

- **quiereVer(contenido)**: Permite representar el conocimiento sobre los contenidos que el usuario quiere ver. Un contenido evalúa a cierto si y sólo si el usuario quiere ver el contenido. Este predicado es necesario para asegurar que el plan incluya los contenidos que el usuario quiere ver. En el fichero de problema se inicializa el predicado en ciertas variables de tipo contenido, y durante la ejecución no se modifican, ya que es un predicado estático.

- **predecesor(contenido1, contenido2)**: Permite representar el conocimiento de qué contenidos tienen una relación de predecesor. Un par de contenidos evalúan a cierto si y sólo si el primero es un predecesor del segundo. Este predicado es necesario para indicar al planificador que restricciones de predecesor existen. En el fichero de problema se inicializa el predicado en ciertos pares de variables de tipo contenido, y durante la ejecución no se modifican, ya que es un predicado estático.

- **necesarioAgendar(contenido**): Permite representar el conocimiento de cuáles son los contenidos pendientes por incluir en el plan. Un contenido evalúa a cierto si y sólo si el contenido no ha sido incluido aún en el plan y debe incluirse. Esto puede ser o bien porque el usuario indicó que quería visualizar el contenido o bien porque pertenece a una cadena de predecesores. Este predicado es necesario para guiar al planificador sobre cuáles son los contenidos a la espera de ser agendados en el plan. Es un predicado de uso interno del dominio, el fichero de problema no debe inicializarlo. Aparece en los efectos de las acciones, por lo que es un predicado dinámico (su evaluación varía durante la ejecución).

- **agendado(contenido, día)**: Permite representar el conocimiento sobre en qué día ha sido agendado un contenido. Un par de contenido y día evalúa a cierto si y sólo si el contenido ha sido incluido en el plan en ese día. Este predicado es necesario porque permite saber cuáles contenidos ya han sido agendados y también si un contenido con predecesores/paralelos puede ser asignado en función del día en que se asignó sus predecesores/paralelos. Es un predicado de uso interno del dominio, el fichero de problema no debe inicializarlo. Aparece en los efectos de las acciones, por lo que es un predicado dinámico.

- **orden(día)**: Evalúa a un número (utiliza fluentes). Permite representar el conocimiento sobre el orden (primero, segundo, etc.) de los días indicados por el usuario. Este predicado es necesario para poder hacer comparaciones entre días, por ejemplo, para ver si un contenido se está asignando a un día posterior al de su predecesor. En el fichero de problema se inicializa el

predicado en todas las variables de tipo día, y durante la ejecución no se modifica, ya que es un predicado estático.

- **contenidosAgendados(día)**: Evalúa a un número (utiliza fluentes). Permite representar el conocimiento sobre el número de contenidos agendados en un día. Este predicado es necesario para poder equilibrar el número de contenidos agendados a los días. En el fichero de problema se inicializa el predicado en todas las variables de tipo día, y durante la ejecución se modifica para reflejar las asignaciones, por lo que es un predicado dinámico.

- **maxContenidosPorDia**: Evalúa a un número (utiliza fluentes). Permite representar el conocimiento sobre cuál es el máximo de contenidos que se pueden asignar en un día. Este predicado es necesario para garantizar el equilibrio de contenidos asignados a días. En el fichero de problema se inicializa el predicado, y durante la ejecución no se modifica, ya que es un predicado estático.

Operadores

Finalmente, con las variables y los predicados ya definidos, los operadores que permiten resolver el problema son los siguientes:

- **requerimiento(contenido)**: Este operador es necesario para determinar los contenidos que necesariamente se han de incluir en el plan. Esto incluye los que el usuario indica que quiere ver y contenidos que pertenecen a una cadena de prerrequisitos para ver un contenido indicado por el usuario. El operador se aplica o bien cuando el usuario indica que quiere ver ese contenido o bien cuando existe un contenido se tiene que incluir en el plan y el contenido es un predecesor de este.

- **agendar(contenido, día)**: Este operador es necesario para incluir en el plan los contenidos que deben verse, ya sea porque pertenecen a una cadena de precondiciones o porque el usuario quiere verlos. El operador se aplica cuando el contenido está marcado como necesario agendar, el día no está lleno y el contenido no tiene predecesores o estos ya se han incluido en el plan en días anteriores.

Modelado del problema

Para modelar los problemas a resolver, utilizamos ficheros problem.pddl que define los objetos, el estado inicial y el estado final deseado. Estos archivos corresponden a juegos de prueba diseñados para verificar la implementación del nivel básico del dominio. El modelado del problema incluye una representación clara de los objetos involucrados y las relaciones necesarias para generar un plan factible.

Objetos

El modelado del problema parte de la definición de dos tipos principales de objetos que encapsulan la información necesaria para representar tanto los contenidos audiovisuales como los recursos temporales disponibles. Estos objetos permiten identificar las unidades individuales que deben ser planificadas y distribuidas como contenidos, mientras que los recursos temporales permiten asignar estos contenidos a un momento de la semana en específico. Los contenidos audiovisuales representan

películas como por ejemplo: *Capitán América: El primer Vengador*, *Capitana Marvel*, *Super Mario Bros*. Para indicar un momento en la semana podemos utilizar por ejemplo: *lunes*, *martes*, etc.

Estado inicial

El estado inicial refleja los requerimientos del usuario, definidas por el predicado quiereVer, y las restricciones narrativas necesarias para cumplir las condiciones del nivel básico. Este define el punto de partida del problema especificando qué elementos requieren planificación y cuáles son sus condiciones actuales. Este estado incluye información sobre qué contenidos deben incluirse en el plan, qué dependencias existen entre ellos, como por ejemplo, *Capitán América: El primer Vengador* es predecesor de *Capitana Marvel*. Este enfoque asegura que el modelo represente tanto las metas del usuario como las limitaciones narrativas o funcionales asociadas a los contenidos. Además el estado inicial considera la disponibilidad de los recursos temporales permitiendo una planificación adecuada desde un conjunto de días totalmente libres.

Estado final

El estado final representa el objetivo a alcanzar y se define en términos de una planificación completa en la que todos los contenidos que el usuario ha indicado cómo quiere ver han sido asignados a recursos temporales disponibles. Es deber del planificador garantizar que cada contenido cumpla con sus requisitos de predecesores y que todos los objetivos del usuario sean satisfechos manteniendo la consistencia y validez del plan generado.

Juegos de prueba

Juego de prueba a mano

Para este juego de prueba se utilizaron cinco contenidos audiovisuales: *Capitán América: El primer Vengador*, *Capitana Marvel*, *Super Mario Bros*, *El Señor de los Anillos: La guerra de los Rohirrim* y *El Señor de los Anillos: Anillos de Poder*. En la siguiente figura se pueden observar las relaciones de predecesor. Una flecha de un contenido c1 a otro c2 indica que c1 es predecesor de c2. Los contenidos que no tienen predecesores son: *Capitán América: El primer Vengador*, *Super Mario Bros* y *El Señor de los Anillos: La guerra de los Rohirrim*. Los contenidos marcados como objetivos (es decir, aquellos que el usuario quiere ver) están resaltados en color verde: *Capitana Marvel*, *Super Mario Bros* y *El Señor de los Anillos: Anillos de Poder*.

Figura 1: Juego de prueba hecho a mano del nivel básico.

Como recursos temporales se dispone de los siete días de la semana: *lunes, martes, miércoles, jueves, viernes, sábado* y *domingo*. Cada día está inicializado como disponible y tiene un límite máximo de un contenido agendado por día, definido mediante el predicado *maxContenidosPorDia*.

El objetivo del planificador es agendar los contenidos objetivos en días específicos respetando las restricciones de predecesores y cumpliendo con las capacidades máximas diarias.

```
step    0: REQUERIMIENTO SUPERMARIOBROS
        1: REQUERIMIENTO CAPITANAMARVEL
        2: REQUERIMIENTO CAPITANAMERICA_ELPRIMERVENGADOR
        3: AGENDAR CAPITANAMERICA_ELPRIMERVENGADOR LUNES
        4: REQUERIMIENTO ELSENORDELOSANILLOS_ANILLOSDEPODER
        5: REQUERIMIENTO ELSENORDELOSANILLOS_LAGUERRADELOSROHIRRIM
        6: AGENDAR ELSENORDELOSANILLOS_LAGUERRADELOSROHIRRIM
MARTES
        7: AGENDAR CAPITANAMARVEL MIERCOLES
        8: AGENDAR SUPERMARIOBROS JUEVES
        9: AGENDAR ELSENORDELOSANILLOS_ANILLOSDEPODER VIERNES
        10: REACH-GOAL
```

Volcado 1: Pasos tomados por el planificador en el juego de prueba hecho a mano del nivel básico.

Tabla con el plan final por días:

Día	Lunes	Martes	Miércoles	Jueves	Viernes
Contenidos	Capitán América: El primer Vengador	El Señor de los Anillos: La guerra de los Rohirrim	Capitana Marvel	Super Mario Bros	El Señor de los Anillos: Anillos de Poder

Tabla 1: Contenidos asignados a días del juego de prueba hecho a mano del nivel básico.

La solución generada por el planificador cumple con todas las restricciones y objetivos establecidos. Los contenidos objetivos (*Capitana Marvel, Super Mario Bros* y *El Señor de los Anillos: Anillos de Poder*) han sido correctamente agendados en días específicos, respetando sus relaciones de predecesores. Los contenidos *Capitán América: El primer Vengador* y *El Señor de los Anillos: La guerra de los Rohirrim*, aunque no estaban marcados como objetivos, fueron incluidos en el plan porque forman parte de las cadenas de predecesores necesarias para cumplir los objetivos.

Además, el plan respeta el límite de un contenido por día, asegurando que cada recurso temporal se utiliza de manera óptima. Esto valida que el modelo del problema en conjunto con el dominio permite resolver casos del nivel básico de forma eficiente y correcta.

En conclusión, el planificador ha demostrado ser capaz de encontrar un plan válido que satisface las restricciones y los requerimientos, lo que confirma la adecuación del modelo para este nivel.

Hemos hecho este juego de prueba para modelar una situación realista en la que un usuario desea planificar el visionado de varias películas durante sus días libres. El usuario tiene un interés particular en tres contenidos (*Capitana Marvel, Super Mario Bros* y *El Señor de los Anillos: Anillos de Poder*), pero debido a sus dependencias narrativas, necesita incluir también predecesores como *Capitán*

América: El primer Vengador y *El Señor de los Anillos: La guerra de los Rohirrim*. Los días disponibles reflejan una semana estándar con un límite de un contenido por día, replicando restricciones comunes como el tiempo libre limitado por jornada. Este caso permite verificar que el modelo puede satisfacer los intereses del usuario respetando las relaciones narrativas y la organización temporal.

Primer juego de prueba mediante generador

En este juego de pruebas se han generado 7 contenidos, de los cuales se han de ver 4 de ellos.
Podemos comprobar que, de los contenidos que tenemos que ver, dos de ellos tienen predecesor.
Tenemos los siguientes contenidos:

Figura 2: Primer juego de prueba generado aleatorio para el nivel básico.

Tenemos en este caso 3 días para asignar los contenidos, *DAY1, DAY2* y *DAY3*. El volcado que nos da el planificador es el siguiente:

```
step  0: REQUERIMIENTO CONTENT1
      1: REQUERIMIENTO CONTENT2
      2: REQUERIMIENTO CONTENT5
      3: REQUERIMIENTO CONTENT4
      4: AGENDAR CONTENT4 DAY1
      5: REQUERIMIENTO CONTENT7
      6: REQUERIMIENTO CONTENT6
      7: AGENDAR CONTENT6 DAY1
      8: AGENDAR CONTENT1 DAY1
      9: AGENDAR CONTENT2 DAY2
     10: AGENDAR CONTENT5 DAY2
     11: AGENDAR CONTENT7 DAY2
     12: REACH-GOAL
```

Volcado 2: Pasos tomados por el planificador en el primer juego de prueba generado aleatorio del nivel básico.

Puesto como una tabla, la asignación de contenidos a días es la siguiente:

Día	DAY1	DAY2	DAY3
Contenidos	Content4 Content6 Content1	Content2 Content5 Content7	

Tabla 2: Contenidos asignados a días del primer juego de prueba generado aleatoriamente del nivel básico.

Podemos ver como la asignación de contenidos por día es correcta debido a que se cubren los predecesores de cada uno de los contenidos asignados. El generador ha asignado un límite máximo de tres contenidos por día, definido mediante el predicado *maxContenidosPorDia*, por lo que aunque el último día haya quedado vacío, la asignación es correcta.

Este juego de prueba se ha escogido debido a su número elevado de contenidos que, además, tienen el máximo de predecesores posibles.

Segundo juego de prueba mediante generador

En este caso generado solamente tenemos 5 contenidos. De estos cinco solamente hay que ver dos, además únicamente hay un predecesor entre los contenidos.

Figura 3: Segundo juego de prueba generado aleatorio para el nivel básico.

En el ejemplo generado tenemos 5 días, este caso se ha generado para que haya días de sobra para asignar los contenidos, el planificador no debería tener ninguna complicación.

```
step  0: REQUERIMIENTO CONTENT5
      1: REQUERIMIENTO CONTENT2
      2: REQUERIMIENTO CONTENT1
      3: AGENDAR CONTENT1 DAY1
      4: AGENDAR CONTENT2 DAY2
      5: AGENDAR CONTENT5 DAY3
      6: REACH-GOAL
```

Volcado 3: Pasos tomados por el planificador en el segundo juego de prueba generado aleatorio del nivel básico.

Puesto de forma estructurada, esta es la tabla con la asignación de contenidos a días:

Día	DAY1	DAY2	DAY3	DAY4	DAY5
Contenidos	Content1	Content2	Content5		

Tabla 3: Contenidos asignados a días del segundo juego de prueba generado aleatoriamente del nivel básico.

Se puede apreciar que la solución es correcta y cubre los predecesores de los contenidos mostrados, aunque en este caso hubiese solamente uno.

Hemos seleccionado este juego de prueba de manera que hubiera pocos contenidos y bastantes días, para verificar que el planificado trabaja adecuadamente en condiciones de alta cantidad de recursos temporales y pocos contenidos.

Extensión 1

Esta es la segunda iteración de la práctica. La extensión 1 parte del nivel básico. Se nos pide encontrar un plan de visionado donde los contenidos pueden tener hasta N predecesores y no tienen contenidos paralelos. El plan debe permitir cubrir los contenidos objetivos en un encadenamiento de contenidos.

Modelado del dominio

El único cambio respecto al nivel básico es que ahora los contenidos pueden tener más de un predecesor. El conocimiento sobre predecesores se simboliza mediante el predicado **predecesor(contenido1, contenido2)** explicado en los predicados del nivel básico. Observamos que este predicado no se limita a una sola relación de predecesor por contenido. Por ejemplo, se puede representar el capítulo *Sherlock: A study in pink* y el capítulo *Sherlock: The blind banker* como predecesores de la película *Sherlock: The abominable bridge*. Además, los operadores del nivel básico para construir el plan funcionan correctamente con contenidos que tienen más de un predecesor. Es decir, la implementación del dominio del nivel básico escala correctamente sin requerir ninguna modificación y permite cubrir la extensión 1. Por esto, las variables, los predicados y las acciones son las mismas que en el modelado del dominio del nivel básico.

Modelado del problema

Como se ha explicado del dominio, no hay que introducir nuevos predicados. El modelado del problema en la Extensión 1 mantiene la estructura del nivel básico, pero con la posibilidad de incluir contenidos con múltiples predecesores. Por esto, ahora hay la posibilidad de definir varios predecesores en un contenido de la forma: predecesor(contenido1, contenidoX), predecesor(contenido2, contenidoX), etc. El objetivo final sigue siendo el mismo: verificar que todos los contenidos marcados como quiere ver han sido agendados.

Juegos de prueba

Juego de prueba a mano

Para este juego de prueba tendremos como contenido tres películas de *Sonic*, cuatro películas de *Los Juegos del Hambre*, una película de Sherlock Holmes y tres capítulos de la serie de Sherlock Holmes. En la Figura 4 se puede observar las relaciones de predecesor. Una flecha de un contenido c1 a otro c2 indica que c1 es predecesor de c2. Además, los contenidos que el usuario ya ha visto están coloreados de azul y los que quiere ver están coloreados de verde.

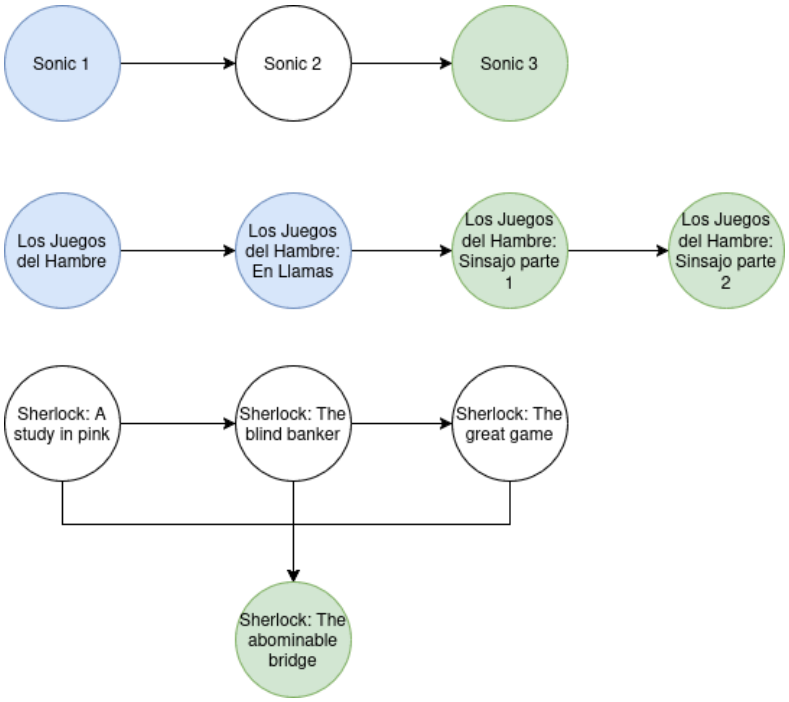

Figura 4: Juego de prueba hecho a mano de la extensión 1.

Como recursos temporales, tendremos los días lunes, martes, miércoles y jueves. Marcaremos un máximo de 2 contenidos por día. El volcado que nos devuelve el programa es el siguiente:

```
step  0: REQUERIMIENTO LOS_JUEGOS_DEL_HAMBRE_SINSAJO_PARTE_1
      1: REQUERIMIENTO LOS_JUEGOS_DEL_HAMBRE_SINSAJO_PARTE_2
      2: REQUERIMIENTO SONIC3
      3: REQUERIMIENTO SONIC2
      4: AGENDAR SONIC2 LUNES
      5: REQUERIMIENTO SHERLOCK_THE_ABOMINABLE_BRIDGE
      6: REQUERIMIENTO SHERLOCK_A_STUDY_IN_PINK
      7: REQUERIMIENTO SHERLOCK_THE_BLIND_BANKER
      8: REQUERIMIENTO SHERLOCK_THE_GREAT_GAME
      9: AGENDAR SHERLOCK_A_STUDY_IN_PINK LUNES
      10: AGENDAR SHERLOCK_THE_BLIND_BANKER MARTES
      11: AGENDAR SHERLOCK_THE_GREAT_GAME MIERCOLES
      12: AGENDAR SONIC3 MARTES
      13: AGENDAR LOS_JUEGOS_DEL_HAMBRE_SINSAJO_PARTE_1 MIERCOLES
      14: AGENDAR LOS_JUEGOS_DEL_HAMBRE_SINSAJO_PARTE_2 JUEVES
      15: AGENDAR SHERLOCK_THE_ABOMINABLE_BRIDGE JUEVES
      16: REACH-GOAL
```

Volcado 4: Pasos tomados por el planificador en el juego de prueba hecho a mano de la extensión 1.

Puesto como tabla por días tenemos:

Día	Lunes	Martes	Miércoles	Jueves
Contenidos	Sonic 2 Sherlock: A study in pink	Sherlock: The blind banker Sonic 3	Sherlock: The great game Los Juegos del Hambre: Sinsajo parte 1	Los Juegos del Hambre: Sinsajo parte 2 Sherlock: The abominable bridge

Tabla 4: Contenidos asignados a días del juego de prueba hecho a mano de la extensión 1.

Observamos que las películas que ya habían sido vistas no se han agendado (aunque eran predecesoras de otras películas). Las películas que querían verse se han agendado correctamente (resaltadas en color verde). Además, en ningún día se agendan más de dos contenidos y todos los contenidos predecesores a un contenido se han asignado en días anteriores. Los contenidos Sonic 2, Sherlock: A study in pink, Sherlock: The blind banker y Sherlock: The great game no están indicados como quiere ver. No obstante, para ver los contenidos objetivo, estos se tienen que añadir, ya que pertenecen a una cadena de predecesores. En conclusión, el planificador ha encontrado un plan que satisface las restricciones y los requerimientos.

Hemos elegido este juego de prueba porque es un ejemplo representativo de un usuario que quiere visualizar una serie de contenidos que están de estreno y que son la continuación de una historia. En este caso, nos encontramos que los contenidos tienen predecesores. En concreto, la película de Sherlock requiere haber visto los primeros tres capítulos de la serie para familiarizarse con los personajes que aparecerán y hay que verificar que el planificador los añada correctamente.

Primer juego de prueba aleatorio

En este juego de prueba tenemos 6 contenidos. Aunque solamente hay un contenido que se debe ver, es un juego de prueba con muchos predecesores. Los contenidos y relaciones del juego de prueba son los siguientes:

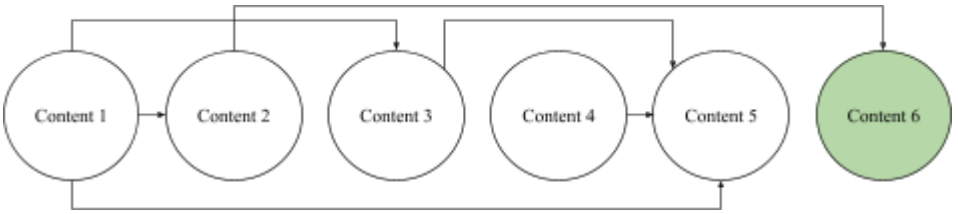

Figura 5: Primer juego de prueba generado aleatorio para la extensión 1.

El generador ha dado como recursos temporales 3 días diferentes. El volcado que nos da el planificador es el siguiente:

```
step     0: REQUERIMIENTO CONTENT6
         1: REQUERIMIENTO CONTENT2
         2: REQUERIMIENTO CONTENT1
         3: AGENDAR CONTENT1 DAY1
         4: AGENDAR CONTENT2 DAY2
         5: AGENDAR CONTENT6 DAY3
```

Volcado 5: Pasos tomados por el planificador en el primer juego de prueba generado aleatorio de la extensión 1.

Puesto como tabla, esta es la asignación de contenidos a días:

Día	DAY1	DAY2	DAY3
Contenidos	CONTENT1	CONTENT2	CONTENT6

Tabla 5: Contenidos asignados a días del primer juego de prueba generado aleatoriamente de la extensión 1.

Se puede observar como en este caso la asignación es correcta, solo había un contenido por ver, pero tenía una cadena de predecesores, de manera que se han tenido que distribuir entre los días anteriores.

Este juego de prueba se ha escogido para comprobar que el planificador administra adecuadamente los contenidos cuando hay muchas relaciones de predecesores entre ellos. En particular, pese a haber muchos predecesores, el planificador ha visto los que eran estrictamente necesarios para que se puedan agendar los contenidos objetivo.

Segundo juego de pruebas aleatorio

Para este caso generado tenemos básicamente lo contrario que en el caso anterior. Hay 7 contenidos, esta vez se han de ver casi todos, pero, sin embargo, hay muy pocos predecesores. En la siguiente figura se pueden observar los contenidos y sus relaciones de predecesor:

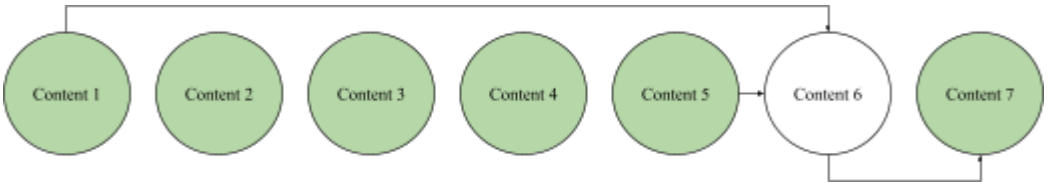

Figura 6: Segundo juego de prueba generado aleatorio para la extensión 1.

En este caso, el generador ha asignado como recursos temporales 5 días. A continuación, este es el volcado con los pasos seguidos por el planificador:

```
step  0: REQUERIMIENTO CONTENT1
      1: REQUERIMIENTO CONTENT2
      2: REQUERIMIENTO CONTENT3
      3: REQUERIMIENTO CONTENT4
      4: REQUERIMIENTO CONTENT5
      5: REQUERIMIENTO CONTENT7
      6: REQUERIMIENTO CONTENT6
      7: AGENDAR CONTENT1 DAY1
      8: AGENDAR CONTENT2 DAY1
      9: AGENDAR CONTENT3 DAY2
      10: AGENDAR CONTENT4 DAY2
      11: AGENDAR CONTENT5 DAY3
      12: AGENDAR CONTENT6 DAY4
      13: AGENDAR CONTENT7 DAY5
      14: REACH-GOAL
```

Volcado 6: Pasos tomados por el planificador en el segundo juego de prueba generado aleatorio de la extensión 1.

Esta es la asignación de contenidos a días visualizada en una tabla:

Día	DAY1	DAY2	DAY3	DAY4	DAY5
Contenido	CONTENT1, CONTENT2	CONTENT3, CONTENT4	CONTENT5	CONTENT6	CONTENT7

Tabla 6: Contenidos asignados a días del segundo juego de prueba generado aleatoriamente de la extensión 1.

Podemos ver como el planificador ha cubierto correctamente todos los predecesores de cada uno de los contenidos, y todos los contenidos por ver se han agendado. Además, podemos comprobar que la asignación ha quedado muy equilibrada entre los diferentes días.

Hemos escogido este caso de prueba debido a que tiene muchos contenidos por ver, y pensamos que este caso extremo sería una buena prueba para comprobar el correcto funcionamiento del planificador.

Extensión 2

Esta es la tercera iteración de la práctica. La extensión 2 parte de la extensión 1. Se nos pide encontrar un plan de visionado donde los contenidos pueden tener hasta N predecesores y hasta M contenidos paralelos. El plan debe permitir cubrir los contenidos objetivos en un encadenamiento de contenidos.

Modelado del dominio

El cambio respecto a la extensión 1 es la introducción de contenidos paralelos. Para poder implementar esta extensión se han tenido que hacer varios cambios respecto a la iteración previa. Recordemos que la extensión 1 utilizaba el modelado del dominio del nivel básico, ya que este escalaba para cubrir la extensión 1 sin requerir cambios.

Variables

Con las variables **día** y **contenido** de la iteración previa es suficiente. No hace falta añadir nuevas variables.

Predicados

Con los mismos predicados que la iteración anterior no podemos representar el conocimiento de qué contenidos son paralelos. En consecuencia, tenemos que añadir el siguiente predicado:

- **paralelo(contenido1, contenido2)**: Permite representar el conocimiento de qué contenidos tienen una relación de simultaneidad. Un par de contenidos evalúan a cierto si y solo si son contenidos paralelos. Este predicado es necesario para indicar al planificador que restricciones de contenidos paralelos existen. En el fichero de problema se inicializa el predicado en ciertos pares de variables de tipo contenido, y durante la ejecución no se modifican, ya que es un predicado estático.

Operadores

Finalmente, tenemos que adaptar los operadores para que cumplan con la condición: Para todos los contenidos del plan se cumple en todo momento que sus paralelos se ven en el mismo día, en el día anterior o en el siguiente. En particular tenemos que hacer las siguientes modificaciones:

- **requerimiento(contenido)**: Como sabemos, este operador determina los contenidos que necesariamente se han de incluir en el plan. Ahora, el operador debe determinar también los contenidos paralelos a contenidos que el usuario quiere ver o que pertenecen a una cadena de predecesores y paralelos. Es decir, se tiene que añadir al operador la nueva condición para aplicarse: si existe un contenido que se tiene que incluir en el plan y el contenido es paralelo a este.

- **agendar(contenido, día)**: Como sabemos, este operador permite incluir en el plan los contenidos que deben verse. Además de las condiciones anteriores para aplicarse (ver agendar en nivel básico), se deben añadir las condiciones:

    - Todos los contenidos paralelos que se han asignado a algún día deben cumplir que este día es el anterior, el mismo o el siguiente al día del operador.

    - Priorizar primero los contenidos que el usuario no ha marcado como quiere ver para asegurar que vea todos los paralelos antes de que el plan finalice su asignación.

Modelado del problema

El único cambio en la definición del dominio que afecta al modelado del problema es la introducción del nuevo predicado para indicar contenidos paralelos. En consecuencia, en el modelado del problema debe indicarse que pares de contenidos son paralelos. Por ejemplo, si un *contenidoX* es paralelo a un *contenidoY*, se debe instanciar el predicado *paralelo(contenidoX, contenidoY)*. Es deber de los operadores agendar los contenidos paralelos en el mismo día o días consecutivos. Finalmente, nuestro objetivo final continúa siendo asegurar que todos los contenidos que se quieren ver han sido agendados.

Juegos de prueba

Juego de prueba a mano

Para este juego de prueba tendremos como contenido dos películas del universo de Marvel y seis capítulos del universo de DC. En concreto, las dos películas de Marvel son paralelas. También, cuatro de los capítulos son paralelos. En la figura 2 se puede observar las relaciones de predecesor y de paralelo. Una flecha de una sola dirección de un contenido c1 a otro c2 indica que c1 es predecesor de c2. Una flecha de doble dirección entre c1 y c2 indica que son contenidos paralelos. Además, los contenidos que el usuario ya ha visto están coloreados de azul y los que quiere ver están coloreados de verde.

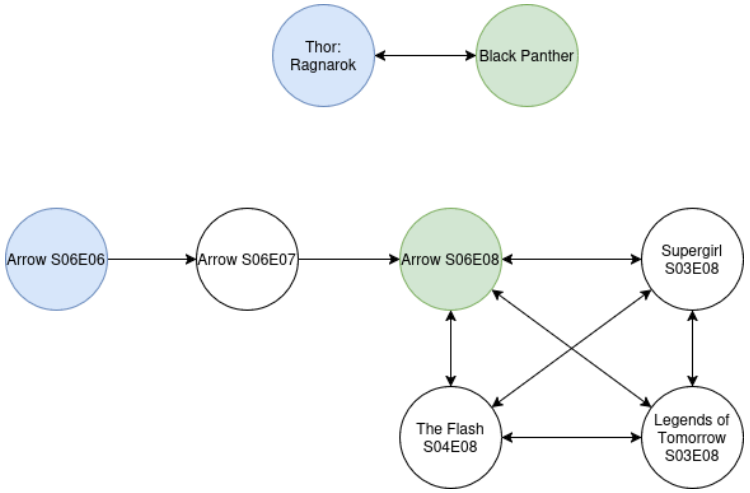

Figura 7: Juego de prueba hecho a mano de la extensión 2.

Como recursos temporales, tendremos los días viernes, sábado y domingo. Marcaremos un máximo de 2 contenidos por día. El volcado que nos devuelve el programa es el siguiente:

```
step  0: REQUERIMIENTO BLACK_PANTHER
      1: REQUERIMIENTO ARROW_S06E08
      2: REQUERIMIENTO ARROW_S06E07
      3: REQUERIMIENTO THE_FLASH_S04E08
      4: REQUERIMIENTO SUPERGIRL_S03E08
      5: REQUERIMIENTO LEGENDS_OF_TOMORROW_S03E08
      6: AGENDAR ARROW_S06E07 VIERNES
      7: AGENDAR THE_FLASH_S04E08 DOMINGO
      8: AGENDAR SUPERGIRL_S03E08 DOMINGO
      9: AGENDAR LEGENDS_OF_TOMORROW_S03E08 SABADO
      10: AGENDAR BLACK_PANTHER VIERNES
      11: AGENDAR ARROW_S06E08 SABADO
      12: REACH-GOAL
```

Volcado 7: Pasos tomados por el planificador en el juego de prueba hecho a mano de la extensión 2.

Puesto como tabla por días tenemos:

Día	Viernes	Sábado	Domingo
Contenidos	Arrow S06E07 Black Panther	Legends of Tomorrow S03E08 Arrow S06E08	The Flash S04E08 Supergirl S03E08

Tabla 7: Contenidos asignados a días del juego de prueba hecho a mano de la extensión 2.

Observamos que los contenidos que ya habían sido vistos no se han agendado. Los contenidos que querían verse se han agendado correctamente (resaltados en color verde). Además, en ningún día se

agendan más de dos contenidos, todos los contenidos predecesores a un contenido se han asignado en días anteriores y todos los contenidos paralelos están dentro de plazo de un día. Los contenidos Legends of Tomorrow S03E08, The Flash S04E08, Supergirl S03E08 no están indicados como quiere ver. No obstante, para ver los contenidos objetivo, estos se tienen que añadir, ya que son contenidos paralelos. En conclusión, el planificador ha encontrado un plan que satisface las restricciones y los requerimientos.

Hemos elegido este juego de prueba porque es un ejemplo representativo de un usuario que quiere visualizar su serie favorita más una peli en el fin de semana. En este caso, nos encontramos con varios contenidos paralelos (esto es frecuente en contenidos de los universos de Marvel y DC) y hay que verificar que el planificador los añada correctamente.

Primer juego de prueba mediante generador

Para este juego de prueba, hemos considerado uno aleatorio que incluye 6 contenidos diferentes. Tres de los cuales están indicados como quiere ver. Además, como se puede observar en la figura 8, este juego de prueba tiene un número de restricciones de contenidos paralelos elevado.

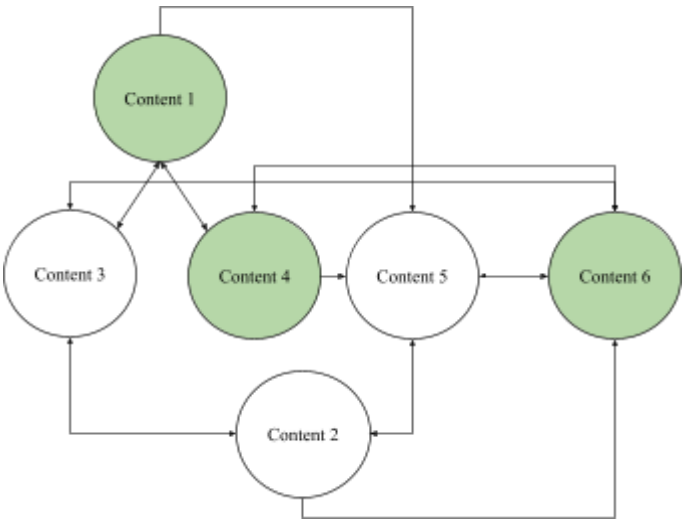

Figura 8: Primer juego de prueba generado aleatorio para la extensión 2.

El generador ha elegido de recursos temporales tres días para asignar los diferentes contenidos. El planificador sigue los siguientes pasos:

```
step  0: REQUERIMIENTO CONTENT4
      1: REQUERIMIENTO CONTENT1
      2: REQUERIMIENTO CONTENT3
      3: REQUERIMIENTO CONTENT2
      4: AGENDAR CONTENT3 DAY1
      5: REQUERIMIENTO CONTENT5
      6: REQUERIMIENTO CONTENT6
      7: AGENDAR CONTENT2 DAY1
      8: AGENDAR CONTENT1 DAY1
      9: AGENDAR CONTENT4 DAY1
      10: AGENDAR CONTENT5 DAY2
      11: AGENDAR CONTENT6 DAY2
      12: REACH-GOAL
```

Volcado 8: Pasos tomados por el planificador en el primer juego de prueba generado aleatorio de la extensión 2.

Así es como quedaría la tabla de asignación de contenidos por día:

Día	DAY1	DAY2	DAY3
Contenido	Content3, Content2, Content4, Content1	Content5, Content6	

Tabla 8: Contenidos asignados a días del primer juego de prueba generado aleatoriamente de la extensión 2.

La planificación es correcta, cubre todos los predecesores y paralelos. De hecho, como hay un número tan elevado de paralelos, ha habido una gran asignación de contenidos a los dos primeros días, debido a que los contenidos se tenían que ver todos el mismo día o con un día de diferencia. Resulta imposible asignar alguno a DAY3.

Hemos elegido este juego de prueba generado aleatoriamente debido a que tenía una cantidad elevada de paralelismos entre contenidos (de hecho todos los contenidos son paralelos), y de esta manera se podría comprobar si el planificador actúa correctamente bajo estas circunstancias extremas. Además, tenemos el doble de contenidos que de días.

Segundo juego de prueba mediante generador

Para este segundo juego de prueba se ha generado un problema aleatorio en el que hay 7 contenidos. Se puede apreciar que hay un gran número de contenidos a ver aunque, sin embargo, en este problema hay pocos predecesores y paralelos en comparación con el anterior. En la siguiente figura se pueden observar los contenidos y sus relaciones:

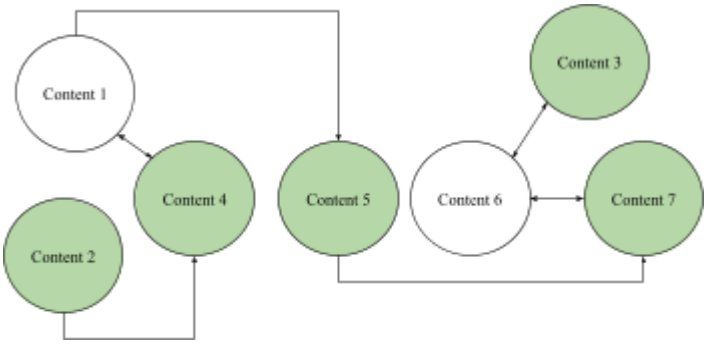

Figura 9: Segundo juego de prueba generado aleatorio para la extensión 2.

El generador ha asignado como recurso temporal cinco días para los siete contenidos. El planificador sigue los siguientes pasos:

```
step  0: REQUERIMIENTO CONTENT7
      1: REQUERIMIENTO CONTENT6
      2: AGENDAR CONTENT6 DAY3
      3: REQUERIMIENTO CONTENT5
      4: REQUERIMIENTO CONTENT1
      5: AGENDAR CONTENT1 DAY1
      6: AGENDAR CONTENT5 DAY2
      7: AGENDAR CONTENT7 DAY3
      8: REQUERIMIENTO CONTENT4
      9: REQUERIMIENTO CONTENT3
      10: AGENDAR CONTENT3 DAY4
      11: REQUERIMIENTO CONTENT2
      12: AGENDAR CONTENT2 DAY1
      13: AGENDAR CONTENT4 DAY2
      14: REACH-GOAL
```

Volcado 9: Pasos tomados por el planificador en el segundo juego de prueba generado aleatorio de la extensión 2.

Representado como una tabla, esta es la asignación de contenidos a días:

Día	DAY1	DAY2	DAY3	DAY4	DAY5
Contenidos	Content1, Content2	Content5, Content4	Content6, Content7	Content3	

Tabla 9: Contenidos asignados a días del segundo juego de prueba generado aleatoriamente de la extensión 2.

Como podemos comprobar, esta solución es correcta y sigue las restricciones impuestas tanto de paralelos como de predecesores. Se puede apreciar que los contenidos se han agendado lo más juntos posible, debido a las presentes cadenas de paralelismos.

Hemos elegido este juego de prueba debido a que hay un equilibrio entre el número de relaciones de predecesores y el número de relaciones de contenidos paralelos. Un caso similar al que ocurre en la realidad. Al ser una distribución común, es adecuada para poner a prueba el planificador.

Extensión 3

Esta es la cuarta iteración de la práctica. La extensión 3 parte de la extensión 2. Se nos pide encontrar un plan de visionado donde se controle que no se añadan más de 3 contenidos por día. El plan debe permitir cubrir los contenidos objetivos en un encadenamiento de contenidos.

Modelado del dominio

El único cambio respecto a la extensión 2 es que ahora no puede haber más de tres contenidos agendados por día. Ya desde el nivel básico, tenemos el predicado maxContenidosPorDia el cual limita el número máximo de contenidos que puede haber en un día cualquiera. Como ahora el máximo se ha fijado a 3, este predicado ya no es necesario. Los únicos cambios que se tendrán que hacer es modificar las comprobaciones del operador agendar(contenido, día) para que verifique que no se están asignando más de tres contenidos en ese día (antes hacía la comprobación con maxContenidosPorDia).

Modelado del problema

Debido a los cambios en el dominio explicados en el apartado anterior, debemos eliminar de nuestro modelado del problema el predicado maxContenidosPorDia, puesto que ahora el número máximo es siempre 3. Ya no debemos preocuparnos por el número de contenidos máximos por día, el planificador se encargará de gestionarlo. El objetivo final sigue siendo la asignación de todos los contenidos que se han de ver.

Juegos de prueba

Juego de prueba a mano

Para este juego de prueba tendremos como contenido dos películas de la saga *Dune* y seis capítulos de un cruce de historias de las series *Chicago Med, Chicago PD* y *Chicago Fire*. En la Figura 10 se pueden observar los contenidos, sus relaciones de predecesor y sus relaciones de paralelo.

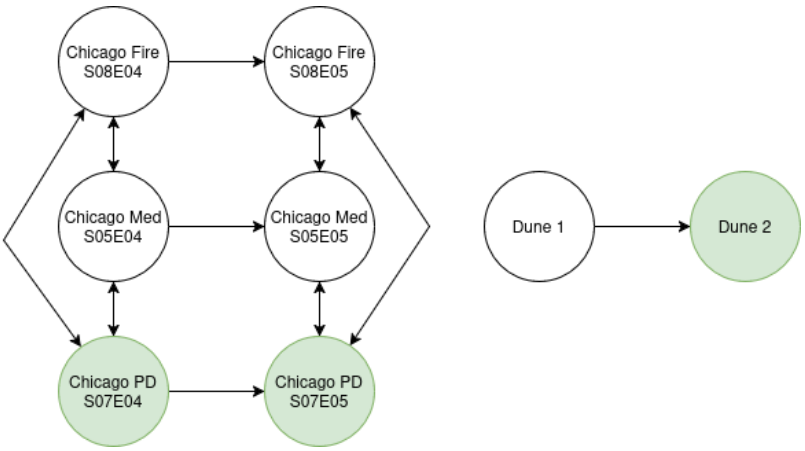

Figura 10: Juego de prueba hecho a mano de la extensión 3.

Como recursos temporales, tendremos los días viernes, sábado y domingo. El volcado que nos devuelve el programa es el siguiente:

```
step  0: REQUERIMIENTO DUNE2
      1: REQUERIMIENTO DUNE1
      2: AGENDAR DUNE1 VIERNES
      3: REQUERIMIENTO CHICAGO_PD_S07E04
      4: REQUERIMIENTO CHICAGO_FIRE_S08E04
      5: REQUERIMIENTO CHICAGO_MED_S05E04
      6: AGENDAR CHICAGO_FIRE_S08E04 VIERNES
      7: AGENDAR CHICAGO_MED_S05E04 SABADO
      8: REQUERIMIENTO CHICAGO_PD_S07E05
      9: REQUERIMIENTO CHICAGO_FIRE_S08E05
     10: REQUERIMIENTO CHICAGO_MED_S05E05
     11: AGENDAR CHICAGO_FIRE_S08E05 DOMINGO
     12: AGENDAR CHICAGO_MED_S05E05 DOMINGO
     13: AGENDAR DUNE2 SABADO
     14: AGENDAR CHICAGO_PD_S07E04 VIERNES
     15: AGENDAR CHICAGO_PD_S07E05 SABADO
     16: REACH-GOAL
```

Volcado 10: Pasos tomados por el planificador en el juego de prueba hecho a mano de la extensión 3.

Puesto como tabla por días tenemos:

Día	Viernes	Sábado	Domingo
Contenidos	Dune 1 Chicago Fire S08E04 Chicago PD S07E04	Chicago Med S05E04 Dune 2 Chicago PD S07E05	Chicago Fire S08E05 Chicago Med S05E05

Tabla 10: Contenidos asignados a días del juego de prueba hecho a mano de la extensión 3.

Las películas que querían verse se han agendado correctamente (resaltadas en color verde). Además, en ningún día se agendan más de tres contenidos, todos los contenidos predecesores a un contenido se han asignado en días anteriores y todos los contenidos paralelos están dentro de un plazo de un día. Observamos cómo pese a haber muchos contenidos paralelos y predecesores, el planificador ha conseguido hacer la planificación sin pasar de los tres contenidos por día. En conclusión, el planificador ha encontrado un plan que satisface las restricciones y requerimientos.

Hemos elegido este juego de prueba porque es un ejemplo representativo de un usuario que quiere visualizar dos capítulos de estreno de su serie favorita más una nueva película. En este caso, nos encontramos que los capítulos pertenecen a una historia entrelazada con capítulos de otras dos series. El objetivo es verificar que el planificador los pueda añadir correctamente sin sobrepasar los tres contenidos por día.

Primer juego de prueba mediante generador

En este caso se genera un problema en el que tenemos 6 contenidos. Se puede observar que todos los contenidos están seleccionados para ser vistos. Además, cabe descartar que hay muy pocos predecesores, y aún menos contenidos paralelos. Esta es la configuración de los contenidos:

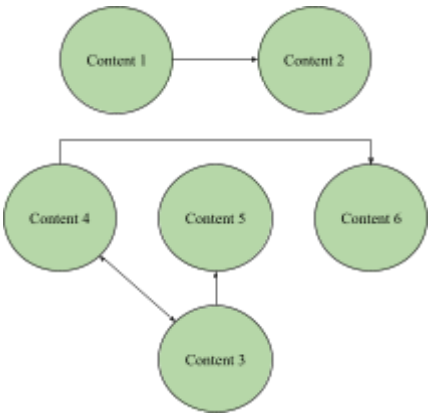

Figura 11: Primer juego de prueba generado aleatorio para la extensión 3.

Los recursos temporales generados son dos días, DAY1 y DAY2. A continuación, estos son los pasos tomados por el planificador para llegar a la solución final.

```
step  0: REQUERIMIENTO CONTENT1
      1: REQUERIMIENTO CONTENT2
      2: REQUERIMIENTO CONTENT3
      3: REQUERIMIENTO CONTENT4
      4: REQUERIMIENTO CONTENT5
      5: REQUERIMIENTO CONTENT6
      6: AGENDAR CONTENT1 DAY1
      7: AGENDAR CONTENT2 DAY2
      8: AGENDAR CONTENT3 DAY1
      9: AGENDAR CONTENT4 DAY1
     10: AGENDAR CONTENT5 DAY2
     11: AGENDAR CONTENT6 DAY2
     12: REACH-GOAL
```

Volcado 11: Pasos tomados por el planificador en el primer juego de prueba generado aleatorio de la extensión 3.

Aquí podemos ver la asignación de los contenidos a los días representada como una tabla:

Día	DAY1	DAY2
Contenidos	Content1, Content3, Content4	Content2, Content5, Content6

Tabla 11: Contenidos asignados a días del primer juego de prueba generado aleatoriamente de la extensión 3.

Como podemos comprobar, el planificador ha asignado los días de una manera en que no hay agendados más de tres contenidos al día. Además de cumplir las restricciones de predecesores y paralelos, de manera que ha encontrado una planificación correcta al problema proporcionado.

Este problema se ha generado con la intención de que hubiese una cantidad justa de días y contenidos, para forzar de esta manera al planificador a repartir los contenidos de la manera más adecuada para no romper la regla de que no haya más de tres contenidos por día. Además, justamente se ha generado un caso en que todos los contenidos se han de ver, de manera que se obliga al planificador a que agende los contenidos.

Segundo juego de prueba mediante generador

Este caso es bastante más complicado que el anterior, aunque tenemos pocas relaciones de paralelismos y predecesores, al tener 9 contenidos se dificulta bastante nuestro problema. Esta es la relación de los contenidos:

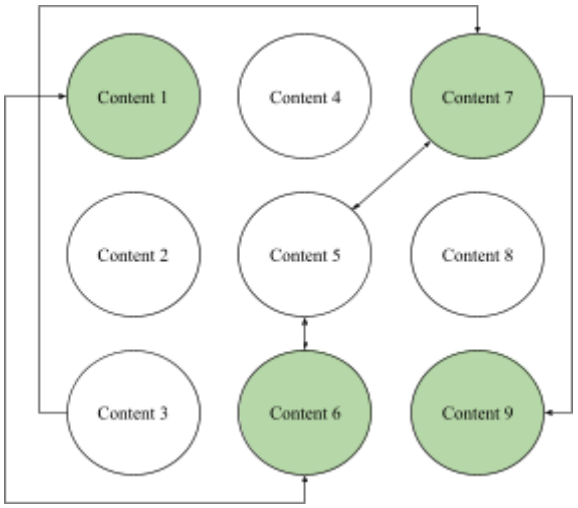

Figura 12: Segundo juego de prueba generado aleatorio para la extensión 3.

En este caso, los recursos temporales generados son tres días: DAY1, DAY2 y DAY3. A continuación, estos son los pasos tomados por el planificador para llegar a la solución final:

```
step  0: REQUERIMIENTO CONTENT1
      1: REQUERIMIENTO CONTENT6
      2: REQUERIMIENTO CONTENT5
      3: REQUERIMIENTO CONTENT7
      4: REQUERIMIENTO CONTENT3
      5: AGENDAR CONTENT3 DAY1
      6: AGENDAR CONTENT5 DAY1
      7: REQUERIMIENTO CONTENT9
      8: AGENDAR CONTENT1 DAY1
      9: AGENDAR CONTENT6 DAY2
     10: AGENDAR CONTENT7 DAY2
     11: AGENDAR CONTENT9 DAY3
     12: REACH-GOAL
```

Volcado 12: Pasos tomados por el planificador en el segundo juego de prueba generado aleatorio de la extensión 3.

Aquí podemos ver la asignación de los contenidos a los días en una tabla:

Día	DAY1	DAY2	DAY3
Contenidos	Content3, Content5, Content1	Content6, Content7	Content9

Tabla 12: Contenidos asignados a días del segundo juego de prueba generado aleatoriamente de la extensión 3.

Podemos ver que en efecto la solución al problema es correcta debido a que no supera los tres contenidos por día y cumple la correcta asignación de contenidos paralelos y predecesores.

Al igual que en el caso anterior, este problema se ha generado con la intención de que hubiese una cantidad justa de días y contenidos. En este caso no se han de ver todos los contenidos, al contrario que en el caso anterior, pero de esta manera tenemos una ejecución más similar a un caso real.

Extensión 4

Esta es la quinta y última iteración de la práctica. La extensión 4 parte de la extensión 2. Se nos pide encontrar un plan de visionado donde se controle que los contenidos agendados en un día no superen los 200 minutos.

Modelado del dominio

El cambio respecto a la extensión 2 es que ahora el límite de asignación a un día está controlado por minutos, mientras que antes era por número de contenidos. Por lo tanto, hay que introducir los minutos en el dominio. Para hacerlo, tenemos que hacer algunos ajustes en las características del dominio.

Variables

Con las variables **día** y **contenido** de la extensión 2 es suficiente. No hace falta añadir nuevas variables.

Predicados

Con los mismos predicados que en la extensión 2 no podemos representar el conocimiento de cuántos minutos dura un contenido. En consecuencia, tenemos que añadir el siguiente predicado:

- **duracion(contenido)**: Evalúa a un número (utiliza fluentes). Permite representar el conocimiento de cuál es la duración en minutos de un contenido cualquiera. Este predicado es necesario para que el planificador pueda controlar el número de minutos agendados en un día. En el fichero de problema se inicializa el predicado en todas las variables de tipo contenido, y durante la ejecución no se modifica, ya que es un predicado estático.

Por otro lado, el predicado contenidosAgendados(día) ya no lo vamos a utilizar porque no queremos saber cuantos contenidos hay agendados en un día, sinó cuánto de lleno en minutos está un día. Es por esto que este predicado se ha reemplazado por:

- **minutosAgendados(día)**: Evalúa a un número (utiliza fluentes). Permite representar el conocimiento sobre el número de minutos agendados en un día como consecuencia de agendar contenidos. Este predicado es necesario para poder cumplir con el límite de 200 minutos por día. En el fichero de problema se inicializa a cero el predicado en todas las variables de tipo día, y durante la ejecución se modifica para reflejar las asignaciones, por lo que es un predicado dinámico.

Finalmente, el predicado maxContenidosPorDia ya no es necesario y podemos eliminarlo. No es necesario añadir un predicado como máximo de minutos por día porque este está fijado globalmente a 200 minutos.

Operadores

Los únicos cambios que se tendrán que hacer es modificar las comprobaciones del operador agendar(contenido, día) para que verifique que no se están asignando más de 200 minutos de contenidos en ese día.

Modelado del problema

Los cambios que se han de hacer en nuestro modelado del problema es cambiar las inicializaciones de los predicados de contenidosAgendados por el de minutosAgendados en cada uno de los días. Además, para cada uno de los contenidos que añadamos en el problema, tendremos también que asignarles una duración usando el predicado duración(contentX). El resto del modelado del problema permanece igual que en la extensión 2.

Juegos de prueba

Juego de prueba a mano

Para este juego de prueba tendremos como contenido cuatro películas y cinco capítulos de la serie *Breaking Bad*. En la Figura 13 se pueden observar los contenidos junto con sus relaciones. Solo el primer episodio de *Breaking Bad* ha sido visto por el usuario.

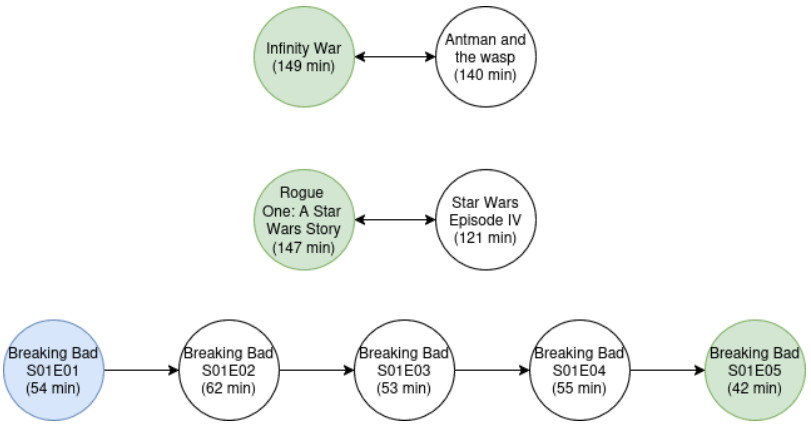

Figura 13: Juego de prueba hecho a mano de la extensión 4.

Como recursos temporales, tendremos los días viernes, sábado, domingo y lunes. Fijémonos que solo el episodio S01E05 de Breaking Bad se puede agendar al mismo día que *Infinity War* sin superar los 200 minutos. Después, solo el episodio S01E03 de *Breaking Bad* se puede agendar al mismo día que Rogue One y juntos dan exactamente 200 minutos. A continuación, solo el episodio S01E04 se puede juntar con Antman and the Wasp. El volcado que nos devuelve el programa es el siguiente:

```
step  0: REQUERIMIENTO INFINITY_WAR
      1: REQUERIMIENTO ANTMAN_AND_THE_WASP
      2: REQUERIMIENTO ROGUE_ONE_A_STAR_WARS_STORY
      3: REQUERIMIENTO STAR_WARS_EPISODE_IV
      4: AGENDAR STAR_WARS_EPISODE_IV VIERNES
      5: REQUERIMIENTO BREAKING_BAD_S01E05
      6: REQUERIMIENTO BREAKING_BAD_S01E04
      7: REQUERIMIENTO BREAKING_BAD_S01E03
      8: REQUERIMIENTO BREAKING_BAD_S01E02
      9: AGENDAR BREAKING_BAD_S01E02 VIERNES
     10: AGENDAR BREAKING_BAD_S01E03 SABADO
     11: AGENDAR BREAKING_BAD_S01E04 DOMINGO
     12: AGENDAR ROGUE_ONE_A_STAR_WARS_STORY SABADO
     13: AGENDAR ANTMAN_AND_THE_WASP DOMINGO
     14: AGENDAR INFINITY_WAR LUNES
     15: AGENDAR BREAKING_BAD_S01E05 LUNES
     16: REACH-GOAL
```

Volcado 13: Pasos tomados por el planificador en el juego de prueba hecho a mano de la extensión 4.

Puesto como tabla por días tenemos:

Día	Viernes	Sábado	Domingo	Lunes
Contenidos	Star Wars Episode IV (121 min) Breaking Bad S01E02 (54 min)	Breaking Bad S01E03 (53 min) Rogue One: A Star Wars Story (147 min)	Breaking Bad S01E04 (55 min) Antman and the wasp (140 min)	Infinity War (149 min) Breaking Bad S01E05 (42 min)
Suma minutos	175 minutos	200 minutos	195 minutos	191 minutos

Tabla 13: Contenidos asignados a días del juego de prueba hecho a mano de la extensión 4.

Las películas que querían verse se han agendado correctamente (resaltadas en color verde). Además, en ningún día se superan los 200 minutos de contenidos, todos los contenidos predecesores a un contenido se han asignado en días anteriores y todos los contenidos paralelos están dentro de un plazo de un día. Observamos que hay muchas combinaciones que sobrepasan los minutos, pero el planificador ha conseguido encontrar la óptima. En conclusión, el planificador ha encontrado un plan que satisface las restricciones y los requerimientos.

Hemos elegido este juego de prueba porque es un ejemplo donde de todas las posibles combinaciones de planes solo una cumple con el máximo de minutos por día. Por lo que se pone a prueba el planificador en un ejercicio complicado. Además, es un juego de prueba representativo con predecesores y paralelos, que podría darse en la realidad. Así pues, el objetivo es que el planificador siga cumpliendo con las restricciones heredadas de las iteraciones pasadas y además cumpla la nueva restricción de máximos minutos por día.

Primer juego de prueba mediante generador

Este problema generado tiene 13 contenidos y muchas relaciones de predecesores y paralelos, de manera que es muy complejo, ideal para poner a prueba el planificador. Aunque solamente hay 5 contenidos por ver, debido a las dependencias, es necesario agendar la gran mayoría. Además, se puede apreciar que la duración de cada uno de los contenidos es muy baja, por lo tanto, cabrán muchos contenidos en un solo día. Los contenidos y su duración son los siguientes:

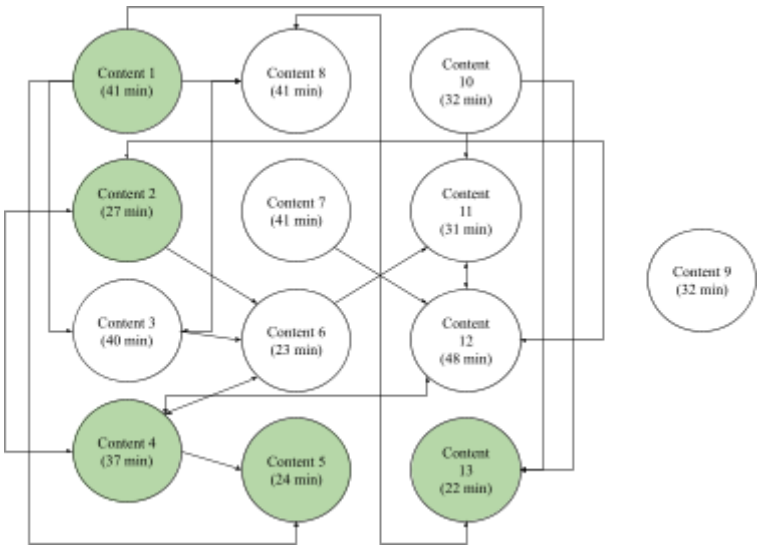

Figura 14: Primer juego de prueba generado aleatorio para la extensión 4.

En este caso, los recursos temporales generados son tres días: DAY1, DAY2 y DAY3. A continuación, los pasos tomados por el planificador para llegar a la solución final:

```
step    0: REQUERIMIENTO CONTENT1
        1: REQUERIMIENTO CONTENT13
        2: REQUERIMIENTO CONTENT10
        3: REQUERIMIENTO CONTENT8
        4: REQUERIMIENTO CONTENT3
        5: AGENDAR CONTENT10 DAY1
        6: REQUERIMIENTO CONTENT2
        7: REQUERIMIENTO CONTENT4
        8: REQUERIMIENTO CONTENT12
        9: REQUERIMIENTO CONTENT7
       10: AGENDAR CONTENT7 DAY1
       11: AGENDAR CONTENT12 DAY2
       12: REQUERIMIENTO CONTENT6
       13: REQUERIMIENTO CONTENT5
       14: AGENDAR CONTENT1 DAY1
       15: AGENDAR CONTENT3 DAY2
       16: AGENDAR CONTENT8 DAY2
       17: AGENDAR CONTENT2 DAY1
       18: AGENDAR CONTENT6 DAY3
       19: AGENDAR CONTENT4 DAY2
       20: AGENDAR CONTENT5 DAY3
       21: AGENDAR CONTENT13 DAY3
       22: REACH-GOAL
```

Volcado 14: Pasos tomados por el planificador en el primer juego de prueba generado aleatorio de la extensión 4.

En esta tabla se puede ver la asignación de los contenidos a los días:

Día	DAY1	DAY2	DAY3
Contenidos	Content10 (32min), Content7 (42min), Content1 (41min), Content2 (27min)	Content12 (48min), Content3 (40min), Content8 (41min), Content4 (37min)	Content6 (23min), Content5 (24min), Content13 (22min)
Suma minutos	142 minutos	166 minutos	69 minutos

Tabla 14: Contenidos asignados a días del primer juego de prueba generado aleatoriamente de la extensión 4.

Podemos apreciar como en este caso, con tantos contenidos de poca duración, el planificador ha ofrecido una planificación correcta del problema, en la que cada día no supera los 200 minutos de visualización y se cumplen las restricciones de paralelos y predecesores.

Hemos elegido este juego de prueba debido al gran número de contenidos que son de poca duración. Para hacerlo se ha puesto una cota inferior y superior a la duración de contenidos. Al además incluir relaciones de predecesores y paralelos, este juego de prueba verifica que el planificador funcione correctamente en casos extremos de muchos contenidos cortos y relacionados.

Segundo juego de prueba mediante generador

Este caso es más simple que el anterior, tenemos 6 contenidos y menos relaciones de predecesores y paralelos. Solo hay por asignar dos contenidos, y como se puede observar, la duración de cada uno de nuestros contenidos es mucho más elevada que en el caso anterior, de forma que lo más normal es que no quepan dos contenidos al día. En la siguiente figura se pueden observar los contenidos con su duración y sus relaciones:

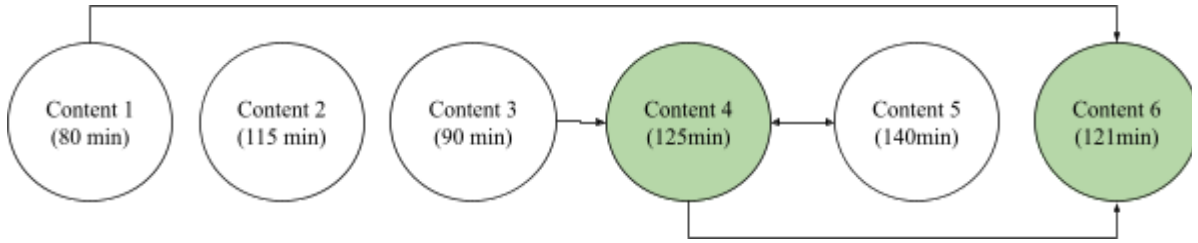

Figura 15: Segundo juego de prueba generado aleatorio para la extensión 4.

En este caso, los recursos temporales generados son cuatro días: DAY1, DAY2, DAY3 y DAY4. A continuación, los pasos tomados por el planificador para llegar a la solución final:

```
step     0: REQUERIMIENTO CONTENT4
         1: REQUERIMIENTO CONTENT3
         2: REQUERIMIENTO CONTENT5
         3: AGENDAR CONTENT3 DAY1
         4: AGENDAR CONTENT5 DAY2
         5: REQUERIMIENTO CONTENT6
         6: REQUERIMIENTO CONTENT1
         7: AGENDAR CONTENT1 DAY1
         8: AGENDAR CONTENT4 DAY3
         9: AGENDAR CONTENT6 DAY4
        10: REACH-GOAL
```

Volcado 15: Pasos tomados por el planificador en el segundo juego de prueba generado aleatorio de la extensión 4.

Representado en forma de tabla, aquí se puede ver la asignación de contenidos a días:

Día	DAY1	DAY2	DAY3	DAY4
Contenidos	Content3 (90 min), Content1 (80 min)	Content5 (140min)	Content4 (125min)	Content6 (121min)
Suma minutos	170 minutos	140 minutos	125 minutos	121 minutos

Tabla 15: Contenidos asignados a días del segundo juego de prueba generado aleatoriamente de la extensión 4.

Se puede apreciar como la solución proporcionada por el planificador es correcta, se cumplen las respectivas dependencias por predecesores y paralelos, a la vez que no se supera el límite de 200 minutos de visualización diarios.

En este caso hemos asignado una cota inferior y superior al número de minutos por contenido, para obtener contenidos de larga duración. Así pues, este juego de prueba verifica el otro extremo respecto al ejemplo anterior: pocos contenidos de larga duración.