PRÁCTICA DE PLANIFICACIÓN

INTELIGENCIA ARTIFICIAL

UNIVERSIDAD POLITÉCNICA DE CATALUÑA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Víctor Moreno Villanueva

Natasha Trojan Jiménez

Julia Gallo Escudero

Índice

1. INTRODUCCIÓN

En un mundo donde el consumo de contenido audiovisual crece exponencialmente, la personalización de la experiencia del usuario se ha convertido en un factor clave para diferenciarse en el mercado. La empresa REDFLIX, reconocida por su variado catálogo de series y películas, busca ir un paso más allá al ofrecer a sus usuarios una herramienta innovadora que les permita planificar el visionado de sus contenidos preferidos de manera eficiente y adaptada a sus necesidades específicas.

Este proyecto tiene como objetivo principal desarrollar una aplicación capaz de generar un plan de visionado diario personalizado. La herramienta debe considerar, por un lado, los contenidos que el usuario ya ha visto y, por otro, aquellos que desea ver. Además, debe respetar las dependencias narrativas entre contenidos, como los predecesores (episodios o películas que deben verse antes para comprender la historia) y los paralelos (contenidos que ocurren en un mismo universo y cuyo orden relativo es relevante para la experiencia narrativa). Todo esto debe combinarse con una distribución equilibrada del tiempo diario, manteniendo un máximo de 200 minutos de visionado por jornada.

El desafío no solo radica en cumplir estas restricciones, sino también en garantizar que el plan final ofrezca la mejor experiencia posible para el usuario. Esto implica:

- Generar un flujo lógico de visionado que respete las relaciones narrativas entre los contenidos.
- Optimizar la cantidad de minutos de visionado por día, asegurando que sea práctico y ajustado a la disponibilidad del usuario.
- Asegurar que los contenidos paralelos y predecesores se incluyan en el plan de manera coherente.

En esta documentación se detalla el proceso de diseño e implementación de la herramienta, desde la estructura de los datos y los algoritmos empleados, hasta las decisiones clave que permiten cumplir con los requisitos planteados. Se aborda el diseño del modelo de dependencias mediante grafos, la ordenación de contenidos basada en relaciones narrativas y la generación del plan diario equilibrado. Todo ello con un enfoque orientado a maximizar la experiencia del usuario, garantizando que pueda disfrutar de los contenidos en las mejores condiciones posibles.

Este trabajo no solo representa un avance en la personalización del consumo de contenido, sino que también sienta las bases para futuras innovaciones en la planificación inteligente, elevando el estándar de lo que los servicios de streaming pueden ofrecer.

2. DOMINIO

2.1. Variables

Las variables en este código son los parámetros que se utilizan en las funciones, predicados y acciones. En este caso, tenemos las siguientes:

- ***?contenido***: Representa un objeto de tipo "contenido", que podría ser una película, un episodio de una serie, un artículo, etc.
- ***?dia****:* Representa un objeto de tipo "día", que puede ser utilizado para asignar un día en el que un contenido será visto.
- ***?pre****:* Un parámetro utilizado en las acciones y predicados para referirse a otro "contenido", específicamente un "contenido predecesor".
- ***?par***: Otro parámetro para referirse a un "contenido paralelo", que es un contenido relacionado con el principal, pero que puede ser visto al mismo tiempo o en paralelo.

2.2. Funciones

Las funciones son aquellas que devuelven un valor, generalmente numérico o booleano. En este caso, tenemos dos funciones:

- ***numero_dia***: Esta función toma un parámetro de tipo dia y devuelve un valor numérico que probablemente representa el número del día en el calendario. Por ejemplo, si el día es el primero del mes, podría devolver 1.

```
(numero_dia ?dia - dia)
```

- ***planificado_para***: Esta función toma un parámetro de tipo contenido y devuelve el día en que ese contenido está planificado para ser visto.

```
(planificado_para ?contenido - contenido)
```

2.3. Predicados

Los predicados son condiciones que se pueden comprobar como verdaderas o falsas. Aquí tenemos varios predicados que representan estados o relaciones entre los contenidos:

- ***(visto ?contenido - contenido)***: Indica si un contenido ha sido visto por el usuario o no.
- ***(predecesor ?contenido - contenido ?pre - contenido)***: Relaciona dos contenidos, indicando que un contenido (el segundo) es el predecesor de otro. Esto puede ser útil si, por ejemplo, un contenido debe verse antes que otro.
- ***(paralelo ?contenido - contenido ?par - contenido):*** Indica que dos contenidos pueden verse al mismo tiempo, es decir, son contenidos paralelos.
- ***(quiere_ver ?contenido - contenido):*** Expresa si el usuario desea ver un contenido en particular.

2.4. Acciones

Las acciones definen lo que sucede cuando se ejecutan. En este dominio, hay tres acciones principales: **querer_predecesor**, **querer_paralelo**, y **ver**.

- ***querer_predecesor***: Esta acción indica que el usuario quiere ver el contenido que es predecesor de otro contenido.
    - **Parámetros**: Un contenido y su predecesor.
    - **Precondiciones**: El usuario no quiere ver el predecesor, pero sí el contenido; además, debe existir una relación de predecesor entre ambos contenidos.
    - **Efectos**: Después de ejecutar la acción, el usuario comienza a querer ver el contenido predecesor.
- ***querer_paralelo***: Similar a la anterior, pero para contenidos que se pueden ver de manera paralela.
    - **Parámetros**: Un contenido y su contenido paralelo.
    - **Precondiciones**: El usuario no quiere ver el contenido paralelo, pero sí el contenido principal; debe existir una relación de paralelismo entre ambos contenidos.
    - **Efectos**: El usuario empieza a querer ver el contenido paralelo.
- ***ver***: Esta es la acción principal en la que el usuario ve un contenido en un día específico.
    - **Parámetros**: Un contenido y el día en que se verá.
    - **Precondiciones**: El usuario quiere ver el contenido, no lo ha visto aún, y las condiciones de los predecesores y contenidos paralelos deben cumplirse (por ejemplo, los contenidos predecesores deben haber sido vistos antes, y los contenidos paralelos deben ser vistos en el mismo día o en días compatibles).

- **Efectos**: El contenido se marca como "visto" y se asigna el día en que fue planificado.

3. MODELIZACIÓN

En este apartado se describen las diferentes extensiones que se han ido realizando donde en cada una de ellas se han ido añadiendo las funcionalidades pedidas.

3.1. Nivel Básico

En este dominio de planificación, el objetivo principal es gestionar el visionado de contenidos, asegurando que los contenidos se ven en una secuencia lógica basada en sus predecesores. Los contenidos pueden tener como máximo un predecesor. El planificador busca organizar el visionado de los contenidos encadenando los elementos de acuerdo a sus predecesores, de forma que cada contenido se vea solo después de haber visto su predecesor, si existe alguno.

Los elementos clave de este dominio son los *contenidos* y los *días*. Los contenidos representan los elementos que el usuario desea ver y los días sirven para asignar cuándo deben ser visualizados. El sistema también utiliza funciones como **`numero_dia`**, que devuelve el día asignado a un contenido, y **`planificado_para`**, que se utiliza para almacenar en qué día se planea ver un contenido específico. Tenemos como ejemplo de predicados **`visto`** indica si un contenido ya ha sido visualizado, **`predecesor`** establece una relación entre dos contenidos donde uno debe verse antes que el otro, y **`quiere_ver`** indica si un contenido está marcado para ser visto.

En cuanto a las acciones disponibles, tenemos tres principales:

1. **`querer_predecesor`**: Esta acción establece que un contenido predecesor que no se quiere ver se marque como objetivo permitiendo que el encadenamiento de contenidos se lleve a cabo correctamente.
2. **`ver_sin_predecesor`**: Esta acción permite ver un contenido si no tiene predecesores, lo que significa que es independiente y puede ser visto directamente. Además, se asigna el día en el que el contenido es visualizado.
3. **`ver_con_predecesor`**: Esta acción se utiliza cuando un contenido tiene un predecesor que ya ha sido visto. Permite ver el contenido solo si el predecesor ya se ha asignado para visualizar.

3.2. Primera Extensión

En esta primera extensión del dominio de planificación de contenidos, se permite que un contenido tenga de 0 a N predecesores, pero sigue sin permitir contenidos paralelos.

En esta versión extendida del dominio de planificación, se añaden las siguientes características sobre la versión anterior:

-Varios predecesores: Los contenidos pueden tener de 0 a N predecesores, en lugar de tener como máximo un predecesor.

-Nueva acción `ver`: Esta acción reemplaza a **`ver_sin_predecesor`** y **`ver_con_predecesor`**, y permite visualizar un contenido solo si todos sus predecesores han sido vistos en días anteriores. Se utiliza un predicado `forall` para garantizar que todos los predecesores hayan sido vistos antes del día asignado al contenido.

3.3. Segunda Extensión

Esta extensión permite que los contenidos puedan tener de 0 a M contenidos paralelos, además de predecesores. El planificador ahora puede gestionar tanto las dependencias de predecesores como los contenidos paralelos. Para cada contenido, todos sus predecesores deben estar visualizados en días anteriores, y sus contenidos paralelos deben ser visualizados el mismo día o en días anteriores.

En esta extensión del dominio, se añaden las siguientes características sobre la versión anterior:

-Contenidos paralelos: Los contenidos ahora pueden tener de 0 a M contenidos paralelos, es decir, pueden existir contenidos que se visualicen simultáneamente. Se introduce el predicado **`paralelo`**, que establece la relación de paralelismo entre los contenidos.

-Nueva acción `querer_paralelo`: Esta acción establece que un contenido paralelo que no se quiere ver se marque como objetivo permitiendo que el encadenamiento de contenidos se lleve a cabo correctamente.

-Modificación de la acción `ver`: Esta acción se amplía para tener en cuenta los contenidos paralelos. Si un contenido tiene contenidos paralelos, todos deben haber sido vistos en días anteriores o el mismo día.

3.4. Tercera Extensión

Este dominio de planificación de contenidos tiene como objetivo controlar la cantidad de contenidos planificados para cada día, asegurando que no se superen los tres contenidos diarios. Además, también gestiona las relaciones entre los contenidos, como las de predecesor y paralelo.

Esta extensión mantiene todas las características de la versión anterior, como la gestión de predecesores y contenidos paralelos, pero añade una restricción importante: limita el número de contenidos que pueden ser visualizados en un mismo día a un máximo de tres. Se introduce una nueva función llamada **`contenidos_planificados`**, que cuenta cuántos contenidos han sido asignados para ser visualizados en un día específico. En la acción **`ver`**, se añade una nueva condición de precondición que verifica que el número de contenidos planificados para un día no supere este límite. Además, cuando un contenido es asignado a un día, la función **`contenidos_planificados`** se incrementa en uno, asegurando que no se exceda el máximo de tres contenidos por día.

3.5. Cuarta Extensión

Esta extensión del dominio introduce una restricción de tiempo para la planificación de contenidos, limitando el total de minutos visualizados en un día a un máximo de 200 minutos.

Para hacer funcionar esta restricción de tiempo, se introducen dos nuevas funciones: **`minutos_consumidos`**, que lleva la cuenta del total de minutos asignados para cada día, y **`duracion`**, que determina cuánto dura un contenido en minutos. En la acción **`ver`**, se agrega una nueva precondición que verifica que el tiempo total planificado para un día, sumando los minutos del contenido que se va a ver, no exceda los 200 minutos. También se añade un efecto en esta acción que actualiza la cantidad de minutos consumidos en un día después de asignar un contenido a ese día. Estas modificaciones permiten que el planificador respete la restricción de tiempo sin sobrepasar el límite de minutos diarios establecidos.

4. INSTANCIAS DEL PROBLEMA

4.1. Instancia base (la estructura se usa para la extensión 0 y 1)

En la **instancia base**, los contenidos se organizan de manera secuencial siguiendo una relación de predecesor, asegurando que ciertos contenidos se visualicen en un orden específico. Además, se define una lista de contenidos que deben verse como objetivo final. Cada contenido se asocia inicialmente con un día sin una planificación específica.

Elementos a inicializar en la instancia base:

- **predecesor:** Define el orden secuencial entre los contenidos.
- **quiere_ver:** Indica los contenidos que deben verse como objetivo final.
- **numero_dia:** Asocia un número específico a cada día disponible.
- **planificado_para:** Inicializa cada contenido sin asignarlo al día 0.

4.2. Instancia de la extensión 2

En la **extensión 2**, se añade la posibilidad de ver contenidos en paralelo, lo que permite que ciertos contenidos puedan ser consumidos simultáneamente. Esta extensión agrega el elemento siguiente:

paralelo: Establece qué contenidos pueden ser vistos simultáneamente.

4.3. Instancia de la extensión 3

La **extensión 3** mantiene las características de la extensión 2, ya que el límite de tres contenidos por día no tiene un impacto directo en esta instancia. Se mantienen las mismas relaciones de paralelismo y la estructura general sin añadir nuevas restricciones.

4.4. Instancia de la extensión 4

En la **extensión 4**, se introduce una restricción adicional relacionada con el tiempo diario disponible. Cada contenido tiene asignada una duración específica, y los días tienen un límite máximo de **200 minutos** para el consumo de contenidos. Además, se inicializan los minutos consumidos por día en **0**.

Nuevos elementos añadidos:

duracion: Define la cantidad de minutos necesarios para completar cada contenido (de 20 a 80 min).

minutos_consumidos: Controla el tiempo total utilizado por día, garantizando que no se supere el límite diario establecido.

5. DESARROLLO DEL PROBLEMA

Para abordar el desarrollo de los distintos problemas, hemos optado por un enfoque incremental basado en prototipos, siguiendo la estructura establecida en el enunciado de la práctica. Comenzamos con el problema más sencillo e incrementamos su complejidad progresivamente, teniendo en cuenta cada una de las extensiones, excepto la extensión 2. A pesar de nuestros intentos, no logramos integrar con éxito en las siguientes extensiones.

En cada prototipo, fuimos incorporando nuevos elementos del problema hasta alcanzar la última extensión. Es importante resaltar que la implementación de cada prototipo se realizó sin necesidad de crear varios prototipos entre las extensiones, ya que la dificultad del problema nos permitió avanzar de una extensión a otra en pocos pasos. Además, hemos creado varios generadores de juegos de prueba para las diferentes extensiones para crear juegos de prueba.

6. JUEGOS DE PRUEBA

Se han creado generadores para diseñar los problemas adecuadamente. Todos los ficheros de problema utilizados se encuentran en la entrega en la carpeta correspondiente al nivel de cada prueba, en la documentación solo se detalla la información relevante.

El **generadorBase** se ha utilizado para las pruebas del nivel básico y para las de la extensión 1 ya que estas no requieren de muchas declaraciones, tan solo de los predecesores. El **generadorExt2** y **generadorExt3** se han utilizado para las extensiones 2 y 3 respectivamente, ya incluyendo los contenidos paralelos, con una restricción de número de contenidos al día añadido al generador3. Por otra parte, el **generadorExt4** se utiliza para realizar los juegos de prueba relacionados con la extensión 4, ya que esta añade un nivel mayor de cambios que requieren otro generador.

Un detalle a tener en cuenta es que los generadores tan solo crean un predecesor para cada contenido que es su contenido inmediatamente inferior. Esto se debe a que, si se hacía de otra forma, corría el riesgo de repetir declaraciones de precedencia o la creación de bucles de precedencia (c2 precedente de c3 y c3 precedente de c2). Para poder comprobar si funciona en todos los casos, se han utilizado los juegos “a mano” para describir situaciones más complejas.

6.1. Nivel básico

En esta ejecución del nivel básico, no hay contenido paralelo y los contenidos tienen solo uno o ningún predecesor. El planificador es capaz de encontrar un plan para poder llegar a ver los contenidos objetivo encadenando contenidos respetando los predecesores. Es importante recalcar que (predecesor c1 c2) indica que c2 es **el predecesor** de c1.

1. Problema “a mano”

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 - contenido
- Init:
    - (predecesor contenido2 contenido1)
    - (predecesor contenido4 contenido3)
    - (quiere_ver contenido2)
    - (quiere_ver contenido4)

OUTPUT:

1. QUERER_PREDECESOR CONTENIDO2 CONTENIDO1
2. QUERER_PREDECESOR CONTENIDO4 CONTENIDO3
3. VER_SIN_PREDECESOR CONTENIDO1 DIA30
4. VER_CON_PREDECESOR CONTENIDO2 CONTENIDO1 DIA31
5. VER_SIN_PREDECESOR CONTENIDO3 DIA30
6. VER_CON_PREDECESOR CONTENIDO4 CONTENIDO3 DIA31
7. REACH-GOAL

Vemos que se respeta el orden y los deseos del usuario y no se ve ningún contenido sin haber visto antes su predecesor.

2. Problema de generador

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 contenido5 contenido6 contenido7 contenido8 contenido9 contenido10 contenido11 contenido12 contenido13 - contenido
- Init:
    - (predecesor contenido6 contenido5)
    - (predecesor contenido12 contenido11)
    - (predecesor contenido9 contenido8)
    - (predecesor contenido4 contenido3)
    - (quiere_ver contenido8)
    - (quiere_ver contenido7)
    - (quiere_ver contenido12)
    - (quiere_ver contenido9)
    - (quiere_ver contenido11)
    - (quiere_ver contenido5)
    - (quiere_ver contenido6)
    - (quiere_ver contenido10)
    - (quiere_ver contenido4)
    - (quiere_ver contenido13)
    - (quiere_ver contenido3)

OUTPUT:

1. VER_SIN_PREDECESOR CONTENIDO3 DIA30
2. VER_CON_PREDECESOR CONTENIDO4 CONTENIDO3 DIA31
3. VER_SIN_PREDECESOR CONTENIDO5 DIA30
4. VER_CON_PREDECESOR CONTENIDO6 CONTENIDO5 DIA31
5. VER_SIN_PREDECESOR CONTENIDO7 DIA31
6. VER_SIN_PREDECESOR CONTENIDO8 DIA30
7. VER_CON_PREDECESOR CONTENIDO9 CONTENIDO8 DIA31
8. VER_SIN_PREDECESOR CONTENIDO10 DIA31
9. VER_SIN_PREDECESOR CONTENIDO11 DIA30
10. VER_CON_PREDECESOR CONTENIDO12 CONTENIDO11 DIA31
11. VER_SIN_PREDECESOR CONTENIDO13 DIA31

De nuevo, la ejecución es correcta y se respetan los deseos y los predecesores.

6.2. Extensión 1

En esta extensión, los contenidos pueden llegar a tener N predecesores. En este caso, se controla no solo que se vean los predecesores de los predecesores, sino también si un contenido tiene más de un predecesor.

1. Problema “a mano”

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 contenido5 - contenido
- Init:
    - (predecesor contenido2 contenido1)
    - (predecesor contenido3 contenido2)
    - (predecesor contenido3 contenido5)
    - (predecesor contenido4 contenido3)
    - (predecesor contenido4 contenido5)
    - (quiere_ver contenido2)
    - (quiere_ver contenido4)

OUTPUT:

1. QUERER_PREDECESOR CONTENIDO4 CONTENIDO5
2. VER CONTENIDO5 DIA1
3. QUERER_PREDECESOR CONTENIDO4 CONTENIDO3
4. QUERER_PREDECESOR CONTENIDO2 CONTENIDO1
5. VER CONTENIDO1 DIA1
6. VER CONTENIDO2 DIA2
7. VER CONTENIDO3 DIA3
8. VER CONTENIDO4 DIA31
9. REACH-GOAL

Tal y como se esperaba, se ha respetado el orden de predecesores: contenido1 → contenido2 → contenido3 → contenido4, y también el hecho de más de un predecesor, ya que antes de ver el contenido4 ve el contenido3 y el contenido5.

2. Problema de generador

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 contenido5 contenido6 contenido7 contenido8 contenido9 - contenido
- Init:
    - (predecesor contenido4 contenido3)
    - (predecesor contenido5 contenido4)
    - (predecesor contenido8 contenido7)
    - (predecesor contenido7 contenido6)
    - (quiere_ver contenido4)
    - (quiere_ver contenido7)
    - (quiere_ver contenido8)
    - (quiere_ver contenido9)
    - (quiere_ver contenido6)
    - (quiere_ver contenido5)

OUTPUT:

1. QUERER_PREDECESOR CONTENIDO4 CONTENIDO3
2. VER CONTENIDO3 DIA1
3. VER CONTENIDO4 DIA2
4. VER CONTENIDO5 DIA31
5. VER CONTENIDO9 DIA31
6. VER CONTENIDO6 DIA1
7. VER CONTENIDO7 DIA2
8. VER CONTENIDO8 DIA31
9. REACH-GOAL

De nuevo, se respetan los órdenes de predecesores y los deseos del usuario.

6.3. Extensión 2

Esta extensión introduce el contenido paralelo, que se puede ver el mismo día o el día anterior al contenido del que es paralelo.

1. Problema "a mano"

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 contenido5 contenido6 contenido7 contenido8 - contenido
- Iinit:
    - (predecesor contenido2 contenido1)
    - (predecesor contenido3 contenido2)
    - (predecesor contenido5 contenido4)
    - (paralelo contenido8 contenido6)
    - (paralelo contenido2 contenido7)
    - (paralelo contenido5 contenido6)
    - (quiere_ver contenido1)
    - (quiere_ver contenido2)
    - (quiere_ver contenido3)
    - (quiere_ver contenido5)

OUTPUT:

1. QUERER_PARALELO CONTENIDO2 CONTENIDO7
1. VER CONTENIDO7 DIA2
2. QUERER_PARALELO CONTENIDO5 CONTENIDO6
3. VER CONTENIDO6 DIA2
4. QUERER_PREDECESOR CONTENIDO5 CONTENIDO4
5. VER CONTENIDO4 DIA1
6. VER CONTENIDO5 DIA2
7. VER CONTENIDO1 DIA1
8. VER CONTENIDO2 DIA2
9. VER CONTENIDO3 DIA31
10. REACH-GOAL

Como es fácil de ver, funciona correctamente respetando el paralelismo.

2. Problema de generador

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 contenido5 contenido6 contenido7 contenido8 contenido9 contenido10 contenido11 contenido12 contenido13 contenido14 - contenido
- Iinit:
    - (predecesor contenido13 contenido14)
    - (predecesor contenido3 contenido4)
    - (predecesor contenido12 contenido13)
    - (paralelo contenido1 contenido7)
    - (paralelo contenido2 contenido4)
    - **(paralelo contenido3 contenido9)**
    - **(paralelo contenido3 contenido14)**
    - (paralelo contenido1 contenido10)
    - (quiere_ver contenido7)
    - **(quiere_ver contenido3)**
    - (quiere_ver contenido9)

OUTPUT:

1. QUERER_PREDECESOR CONTENIDO3 CONTENIDO4
2. QUERER_PARALELO CONTENIDO3 CONTENIDO14
3. VER CONTENIDO4 DIA30
4. VER CONTENIDO7 DIA31
5. **VER CONTENIDO9 DIA31**
6. **VER CONTENIDO14 DIA31**
7. **VER CONTENIDO3 DIA31**
8. REACH-GOAL

Como es fácil de ver, funciona correctamente respetando el paralelismo.

6.4. Extensión 3

Esta extensión añade una restricción de capacidad al problema. Aquí, se busca que el planificador no ponga más de tres contenidos en el mismo día.

1. Problema “a mano”

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 contenido5 - contenido
- Iinit:
    - (predecesor contenido2 contenido1)
    - (predecesor contenido3 contenido2)
    - (predecesor contenido4 contenido3)
    - (predecesor contenido5 contenido4)
    - (quiere_ver contenido1)
    - (quiere_ver contenido2)
    - (quiere_ver contenido4)
    - (quiere_ver contenido5)

El planificador debe asegurarse de no asignar todos al mismo día, ya que las restricciones de predecesores y paralelos no afectan a los que el usuario quiere ver.

OUTPUT:

1. VER CONTENIDO1 DIA1
2. VER CONTENIDO2 DIA2
3. QUERER_PREDECESOR CONTENIDO4 CONTENIDO3
4. VER CONTENIDO3 DIA3
5. VER CONTENIDO4 DIA4
6. VER CONTENIDO5 DIA5
7. REACH-GOAL

Podemos ver que, efectivamente, no asigna todos al mismo día a pesar de poder.

2. Problema de generador

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 contenido5 contenido6 contenido7 contenido8 contenido9 contenido10 contenido11 contenido12 contenido13 contenido14 - contenido
- Iinit:
    - (predecesor contenido7 contenido8)
    - (predecesor contenido6 contenido7)
    - (predecesor contenido5 contenido6)
    - (paralelo contenido4 contenido9)
    - (paralelo contenido5 contenido6)
    - (paralelo contenido9 contenido12)
    - (paralelo contenido7 contenido14)
    - (paralelo contenido8 contenido12)
    - (paralelo contenido5 contenido12)
    - (quiere_ver contenido8)
    - (quiere_ver contenido10)
    - (quiere_ver contenido3)
    - (quiere_ver contenido11)
    - (quiere_ver contenido9)

OUTPUT:

1. QUERER_PARALELO CONTENIDO9 CONTENIDO12
2. VER CONTENIDO3 DIA31
3. VER CONTENIDO10 DIA31
4. VER CONTENIDO11 DIA31
5. VER CONTENIDO12 DIA30
6. VER CONTENIDO8 DIA30
7. VER CONTENIDO9 DIA30
8. REACH-GOAL

No asigna más de tres contenidos a cada día, respetando el límite de capacidad diario.

6.5. Extensión 4

Esta extensión añade una restricción de tiempo, donde los contenidos tienen asignados el número de minutos de duración, y el planificador controla que el plan generado no supere los 200 minutos al día.

1. Problema “a mano”

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 contenido5 contenido6 contenido7 contenido8 contenido9 contenido10 - contenido
- Iinit:
    - (= (duracion contenido1) 25)
    - (= (duracion contenido2) 30)
    - (= (duracion contenido3) 50)
    - (= (duracion contenido4) 35)
    - (= (duracion contenido5) 45)
    - (= (duracion contenido6) 55)
    - (= (duracion contenido7) 40)
    - (= (duracion contenido8) 60)
    - (= (duracion contenido9) 65)

- (= (duracion contenido10) 70)
- (predecesor contenido1 contenido2)
- (predecesor contenido2 contenido3)
- (predecesor contenido4 contenido5)
- (paralelo contenido1 contenido6)
- (paralelo contenido2 contenido7)
- (paralelo contenido5 contenido8)
- (quiere_ver contenido1)
- (quiere_ver contenido2)
- (quiere_ver contenido3)
- (quiere_ver contenido5)

Las duraciones están en un rango de 20 a 80.

OUTPUT:

1. VER CONTENIDO3 DIA1
8. QUERER_PARALELO CONTENIDO5 CONTENIDO8
9. VER CONTENIDO8 DIA1
10. VER CONTENIDO5 DIA1
11. QUERER_PARALELO CONTENIDO2 CONTENIDO7
12. VER CONTENIDO7 DIA2
13. VER CONTENIDO2 DIA2
14. QUERER_PARALELO CONTENIDO1 CONTENIDO6
15. VER CONTENIDO6 DIA3
16. VER CONTENIDO1 DIA3
17. REACH-GOAL

DIA1: c3 + c8 + c5 = 50 + 60 + 45 = 155

DIA2: c7 + c2 = 40 + 30 = 50

DIA3: c6 + c1 = 55 + 25 = 80

Se puede observar que no se asigna contenido a días en los que, si se asignara, sobrepasaría el tiempo de 200 minutos.

2. Problema de generador

INPUT:

- Objects:
    - contenido1 contenido2 contenido3 contenido4 contenido5 contenido6 contenido7 contenido8 contenido9 contenido10 contenido11 contenido12 contenido13 contenido14 - contenido
- Iinit:
    - (= (duracion contenido1) 15)
    - (= (duracion contenido2) 60)
    - (= (duracion contenido3) 75)
    - (= (duracion contenido4) 30)
    - (= (duracion contenido5) 60)
    - (= (duracion contenido6) 75)
    - (= (duracion contenido7) 15)
    - (= (duracion contenido8) 75)
    - (= (duracion contenido9) 75)
    - (= (duracion contenido10) 60)
    - (= (duracion contenido11) 60)
    - (= (duracion contenido12) 75)
    - (= (duracion contenido13) 75)
    - (predecesor contenido12 contenido11)
    - (predecesor contenido4 contenido3)
    - (predecesor contenido5 contenido4)
    - (predecesor contenido10 contenido9)
    - (predecesor contenido7 contenido6)
    - (predecesor contenido13 contenido12)
    - (predecesor contenido6 contenido5)
    - (paralelo contenido6 contenido12)
    - (paralelo contenido2 contenido9)
    - (paralelo contenido1 contenido12)
    - (paralelo contenido5 contenido12)
    - (paralelo contenido7 contenido9)
    - (quiere_ver contenido12)

- (quiere_ver contenido11)
- (quiere_ver contenido9)
- (quiere_ver contenido13)
- (quiere_ver contenido3)
- (quiere_ver contenido8)
- (quiere_ver contenido5)

En el generador, la duración del contenido va de 15 a 90 en incrementos de 15.

OUTPUT:

1. VER CONTENIDO3 DIA1
2. VER CONTENIDO8 DIA1
3. VER CONTENIDO9 DIA2
4. VER CONTENIDO11 DIA2
5. VER CONTENIDO12 DIA3
6. VER CONTENIDO13 DIA4
7. QUERER_PREDECESOR CONTENIDO5 CONTENIDO4
8. VER CONTENIDO4 DIA2
9. VER CONTENIDO5 DIA3
10. REACH-GOAL

DIA1: c3 + c8 = 75 + 75 = 150

DIA2: c9 + c11 + c4 = 75 + 60 + 30 = 165

DIA3: c12 + c5 = 75 + 60 = 135

DIA4: c13 = 75

Se puede observar que no se asigna contenido a días en los que, si se asignara, sobrepasaría el tiempo de 200 minutos.

7. CONCLUSIÓN

En este trabajo, hemos abordado un problema de planificación automatizada utilizando un modelo formal basado en **PDDL (Planning Domain Definition Language)**, lo que nos permitió representar de manera estructurada los distintos elementos, relaciones y restricciones del problema. A lo largo del desarrollo, nos centramos en la asignación eficiente de contenidos que un usuario desea ver, estableciendo un orden lógico a través de relaciones de **predecesor**, la identificación de contenidos clave con **quiere_ver** y la organización temporal mediante **número de día** y **planificado_para**.

El enfoque seguido se desarrolló de manera incremental, comenzando con una **instancia base** y extendiéndose progresivamente con nuevas complejidades, como la posibilidad de ver contenidos en paralelo (**paralelo**) y la introducción de restricciones temporales específicas (**duración** y **minutos_consumidos**). Estas extensiones no solo enriquecieron el modelo, sino que también presentaron desafíos adicionales que requirieron un análisis detallado para garantizar la coherencia y eficiencia de la planificación resultante.

Para la resolución automática de las distintas instancias del problema, utilizamos el planificador **Fast Forward (FF)**, una herramienta reconocida por su eficiencia en la generación de planes óptimos en tiempos de ejecución razonables. A través de este proceso, validamos la capacidad del sistema para adaptarse a distintos niveles de complejidad, ajustándose tanto a las relaciones de dependencia entre contenidos como a las restricciones temporales diarias.

Gracias a esto, pudimos comprobar que este tipo de sistemas es capaz de generar una planificación eficiente para el usuario en tiempos de ejecución razonables.