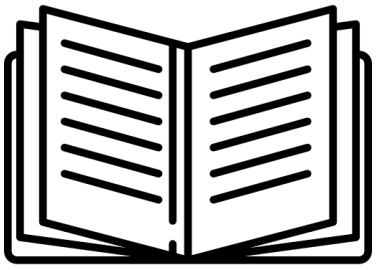
IA - PRÁCTICA 3

PLANIFICACIÓN DE LIBROS



Edgar Perez Blanco

Bartomeu Perelló Comas

Marcos Gómez Vázquez

IA - GEI FIB

Cuatrimestre de Otoño 2019-2020

ÍNDICE

EL PROBLEMA

Nos encontramos con un problema de planificación, en el que después de crear un sistema de recomendación de libros para un cliente, éste nos pide que creemos un sistema capaz de hacer un plan de lectura mensual el cual tiene que indicar, a partir de un conjunto de libros establecido, el orden de lectura de los libros.

Respecto a los libros, hay que tener en cuenta que puede haber algunos que ya hayan sido leídos, y éstos deben ser descartados. También hay libros que tienen predecesores, y lógicamente el plan debe indicar un orden de lectura correcto, indicando que se deben leer los libros predecesores previamente. Además hay libros paralelos, los cuales tienen historias que transcurren una después de la otra y en paralelo, lo que quiere decir que los libros que sean paralelos se deben leer en un orden concreto para poder disfrutar de la experiencia de lectura como debería ser.

La versión final del planificador debe indicar también el número de páginas que se deben leer cada mes, el cual no puede superar las 800 páginas. Esto quiere decir que como información de los libros debe indicarse su número de páginas.

MODELIZACIÓN DEL DOMINIO

Básico + Extensión 1

Cuando empezamos a desarrollar el nivel básico, programamos sin darnos cuenta la extensión 1, es decir, el código que realizamos para el nivel básico resultó funcionar con la extensión 1. Creemos que dado que el básico pedía algo más simple que la extensión posiblemente existe un conjunto de acciones más simple y rápido para solucionar este problema, pero decidimos mantener este porque el planificador nos pareció suficientemente rápido (los resultados eran prácticamente instantáneos). Por lo tanto, detallamos ambos apartados fusionados en uno, dado que el código es exactamente el mismo.

La extensión 1 añade al sistema la posibilidad de que un libro tenga no solo cero o uno sucesores, sino de cero hasta N. Esto implica que para añadir un libro, no tengas que comprobar solo si su predecesor este ya añadido o no (nivel básico) sino que ahora cada libro puede tener múltiples sucesores y esto puede suceder en cadena.

Tipos

Del mismo modo que en el nivel básico, nuestro planificador sólo necesitará dos tipos de objetos:

- **Libro:** Como su nombre indica, cada instancia de este tipo representa un libro que, mediante los predicados, se relaciona con otros libros.

- **Mes:** Referencia temporal utilizada por el sistema para la planificación. En todo problema estándar sólo existirán doce instancias del mismo que harán referencia a los meses reales del año. Si quisiéramos hacer planes de mas meses, solamente sería necesario instanciarlos y relacionarlos temporalmente entre ellos mediante los predicados.

Predicados

Contamos con diferentes tipos de predicados. Algunos están destinados a modelar el escenario en el que trabaja el sistema y otros a facilitar el avance del planificador en su tarea. Para su explicación, los hemos clasificado según el conjunto de restricciones al que están destinados a modelar.

Predicados utilizados para satisfacer las condiciones impuestas por el usuario (objetivo de lectura):

- **(leído ?x - libro)**

  Expresa que el libro X está leído (incluido en el plan). Este es utilizado por el usuario para marcar los libros previamente leídos, y para establecer el objetivo del planificador (los libros que el usuario quiere leer).

Predicados utilizados para satisfacer las condiciones impuestas por la relación entre libros predecesores:

- **(anterior ?x - mes ?y - mes)**

  Expresa que el mes X es estrictamente anterior al mes Y. Este es utilizado para modelar el escenario temporal en el ámbito de los predecesores, ya que el planificador debe planear la lectura de todo libro predecesor a otro, en un mes anterior en el tiempo al de su sucesor.

- **(predecesor ?x - libro ?y - libro)**

  Expresa que el libro X es predecesor del libro Y. Este es utilizado para modelar la relación “predecesor - sucesor” entre libros. En este caso, del mismo modo que en la extensión 1, un libro puede tener de 0 a N predecesores.

- **(predecesor_pendiente ?x - libro ?y - mes)**

  Expresa que el libro X debe ser leído en un mes estrictamente posterior al mes Y. Este es generado por el planificador durante la ejecución cuando un libro es añadido al plan de lectura en el mes Y y este era predecesor de X.

Acciones

- **leer**

Acción que corresponde a leer un libro en un mes concreto. A grandes rasgos la acción controla que no queden predecesores por leer. Si esto pasa, la precondición controla que las restricciones temporales se cumplan con los hechos de tipo “pendiente”.

Estos hechos de tipo pendiente son precisamente creados por el mismo efecto de esta acción. Este elimina la relación de predecesor a cambio de, para cada libro del que este es predecesor y no haya sido leído, guarda qué mes deben respetar. De este modo, todos los libros se deben leer en un mes que respete a todos sus predecesores temporalmente.

La precondición de esta acción se entiende con la siguiente frase:

Un libro L se lee en un mes concreto M si:
- *L no está leído*
- *no existe ningún predecesor no-leído*
- *M es posterior a los meses de lectura de todos los predecesores leídos.*

El efecto de la acción está dividido en dos partes. La primera consiste simplemente en marcar como leído el libro L. La segunda consiste en eliminar la condición de predecesor a otros libros (marcándolo como ya añadido al plan) y añadir para todo sucesor de L un hecho que marque a partir de que mes se podrá leer (hecho predecesor_pendiente).

```
(:action leer
    :parameters (?l - libro ?m - mes)
    :precondition (and
        (not (leido ?l))                                                    ; Libro targeteado no leido.
        (not (exists (?p - libro) (predecesor ?p ?l)))                      ; No hay predecesores sin leer.
        (forall (?t - mes) (imply (predecesor_pendiente ?l ?t) (anterior ?t ?m)))   ; El mes targeteado es anterior a al
                                                                            ; de todos los predecesores leidos.
    )
    :effect (and
        (leido ?l)                                  ; Marcamos el libro como leido.
        (forall (?p - libro)  (when (predecesor ?l ?p)  ; Para todos los libros de los que el libro es predecesor.
            (and
                (not (predecesor ?l ?p))            ; Borramos el statement de ser predecesor.
                (predecesor_pendiente ?p ?m)        ; Establecemos el mes al que deben ser sus sucesores estrictamente posteriores.
            )
        ))
    )
)
```

- **set_leido**

Acción que elimina todas las restricciones de predecesor que tienen los libros ya leídos, dado que estas no interferiran en el plan. Esta acción solo es ejecutada para los libros que el usuario ya se ha leído previamente (libros fuera del plan) ya que los efectos de esta son comunes a la acción leer y esta nunca será necesaria para el planificador.

```
(:action set_leido
    :parameters (?l - libro)
    :precondition (leido ?l)
    :effect (forall (?p - libro) (when (predecesor ?l ?p) (not (predecesor ?l ?p))))
)
```

Necesitamos esta acción para que los libros leídos previamente leídos no interfieran en la planificación.

Si estos eran predecesor de algún libro, o alguien les precede a ellos, al estar leídos, ya cumplen con todas las condiciones. Por lo tanto eliminar esta relacion solo elimina elementos del dominio que facilitaran la búsqueda y simplificarán ligeramente la acción principal de leer un libro.

Extension 2

La extensión 2 añade al sistema la posibilidad de que un libro tenga otros que sean paralelos. Esta relación, al ser simétrica entre libros, dificulta el sistema de las precondiciones, ya que no se puede tener en cuenta sólo un sentido, sino que se deben revisar relaciones en los dos sentidos. En los apartados de predicados y acciones se detalla como se ha solventado este problema.

Tipos

De nuevo, como en las ya descritas anteriores versiones, en esta extensión solo hemos utilizado dos tipos de elementos:

- **Libro:** Como su nombre indica, cada instancia de este tipo representa un libro que, mediante los predicados, se relaciona con otros libros.

- **Mes:** Referencia temporal utilizada por el sistema para la planificación. En todo problema estándar sólo existirán doce instancias del mismo que harán referencia a los meses reales del año. Si quisiéramos hacer planes de mas meses, solamente sería necesario instanciarlos y relacionarlos temporalmente entre ellos mediante los predicados.

Predicados

Contamos con diferentes tipos de predicados. Algunos están destinados a modelar el escenario en el que trabaja el sistema y otros a facilitar el avance del planificador en su tarea. Para su explicación, los hemos clasificado según el conjunto de restricciones al que están destinados a modelar.

Predicados utilizados para satisfacer las condiciones impuestas por el usuario (objetivo de lectura):

- **(leído ?x - libro)**

  Expresa que el libro X está leído (incluido en el plan). Este es utilizado por el usuario para marcar los libros previamente leídos, y para establecer el objetivo del planificador (los libros que el usuario quiere leer).

Predicados utilizados para satisfacer las condiciones impuestas por la relación entre libros predecesores:

- **(anterior ?x - mes ?y - mes)**

  Expresa que el mes X es estrictamente anterior al mes Y. Este es utilizado para modelar el escenario temporal en el ámbito de los predecesores, ya que el planificador debe planear la lectura de todo libro predecesor a otro, en un mes anterior en el tiempo al de su sucesor.

- **(predecesor ?x - libro ?y - libro)**

  Expresa que el libro X es predecesor del libro Y. Este es utilizado para modelar la relación “predecesor - sucesor” entre libros. En este caso, del mismo modo que en la extensión 1, un libro puede tener de 0 a N predecesores.

- **(predecesor_pendiente ?x - libro ?y - mes)**

  Expresa que el libro X debe ser leído en un mes estrictamente posterior al mes Y. Este es generado por el planificador durante la ejecución cuando un libro es añadido al plan de lectura en el mes Y y este era predecesor de X.

Predicados utilizados para satisfacer las condiciones impuestas por la relación entre libros paralelos:

- **(dist01 ?x - mes ?y - mes)**

  Expresa que el mes X está a distancia 0 o 1 del mes Y. Este es utilizado para modelar el escenario temporal en el ámbito de los paralelos, ya que el planificador debe planear la lectura de todo libro paralelo a otro, en un mes igual, directamente anterior o directamente posterior en el tiempo al de su sucesor (distancia 0 o 1).

- **(paralelo ?x - libro ?y - libro)**
  Expresa que el libro X es paralelo al libro Y. Este es utilizado para modelar la relación “predecesor - sucesor” entre libros. En este caso, del mismo modo que en la extensión 1, un libro puede tener de 0 a N predecesores.

- **(paralelos_pendientes ?x - libro ?y - mes)**

  Expresa que el libro X debe ser leído en un mes a distancia 0 o 1 del mes Y. Este es generado por el planificador durante la ejecución cuando un libro es añadido al plan de lectura en el mes Y y este era paralelo a X.

- **(end)**

  Predicado de control que provoca que el sistema no pueda terminar un plan sin haber revisado la existencia de libros paralelos a los incluidos en el plan. Una vez este predicado se vuelve cierto, el sistema se ve obligado a terminar su ejecución, ya que su no - existencia es precondición de todas las acciones y además, siempre formará parte del objetivo del usuario (debe ser especificado como parte del objetivo en el problema).

Acciones

- **leer**

Acción que corresponde a leer un libro en un mes concreto. A grandes rasgos la acción controla que no queden predecesores por leer ni paralelos en una dirección por leer. Si esto pasa, la precondición controla que las restricciones temporales se cumplan con los hechos de tipo “pendiente”.

Estos hechos de tipo pendiente son precisamente creados por el mismo efecto de esta acción. Este elimina la relación de predecesor y de paralelo a cambio de, para cada libro del que es predecesor o paralelo y no haya sido leído, guarda para todos ellos qué mes deben respetar. De este modo, todos los libros se deben leer en un mes que respete a todos sus predecesores temporalmente.

```
(:action leer
    :parameters (?l - libro ?m - mes)
    :precondition (and

        (not (end))                                             ; El programa no ha acabado.
        (not (leido ?l))                                        ; Libro targeteado no leido.
        (not (exists (?p - libro) (predecesor ?p ?l)))          ; No hay predecesores sin leer.
        (not (exists (?p - libro) (paralelo ?p ?l)))            ; No hay libros paralelos a mi (en este sentido) sin leer.


        (forall (?t - mes) (and                                 ; Comprobacion de fechas
            (imply (predecesor_pendiente ?l ?t) (anterior ?t ?m))   ; El mes es anterior a al de todos los predecesores leidos.
            (imply (paralelos_pendientes ?l ?t) (dist01 ?t ?m))     ; El mes esta a distancia (0,1) de todos los libros que har
        ))

    )
    :effect (and

        (leido ?l)                                              ; Marcamos el libro como leido.

        (forall (?mm - mes) (not (paralelos_pendientes ?l ?mm)))    ; Eliminamos el libro sujeto de la 'lista' de paralelos per

        (forall (?p - libro)
            (when (predecesor ?l ?p) (and
                (not (predecesor ?l ?p))                        ; Borramos la restriccion de ser predecesor (ya que este es
                (predecesor_pendiente ?p ?m)                    ; Establecemos el mes al que sus sucesores deben ser estric
            ))
        )

        (forall (?p - libro)
            (when (paralelo ?l ?p) (and
                (not (paralelo ?l ?p))                          ; Borramos la restriccion de ser paralelo (ya que este esta
                (paralelos_pendientes ?p ?m)                    ; Establecemos el mes al que sus paralelos deben estar a di
            ))
        )
    )
)
```

La precondición de esta acción se entiende con la siguiente frase:

Un libro L se lee en un mes concreto M si:

- *L no está leído*
- *no existe ningún predecesor no-leído*
- *no existe ningún libro paralelo* a L no-leído.*
- *M es posterior a los meses de lectura de todos los predecesores*
- *M está a distancia 0 o 1 de todos los meses de lectura de los libros paralelos a L ya leídos.*

* Libros paralelos solo en el sentido otro libro es paralelo al que se va a leer.

El efecto de la acción está dividido en 4 partes:

1. La primera consiste simplemente en marcar como leído el libro L.

2. La segunda es eliminar todos los hechos que referencian al libro L como libro paralelo pendiente de lectura (ha habido otros libros paralelos a este que ya han sido añadidos al plan y han marcado a este como paralelo pendiente).

3. La tercera consiste en eliminar la condición de predecesor a otros libros (marcándolo como ya añadido al plan) y añadir para todo sucesor de L un hecho que marque a partir de que mes se podrá leer (hecho predecesor_pendiente).

4. Y la cuarta y última parte consiste eliminar la condición de paralelo en el sentido contrario al inspeccionado por la precondición (es aquí donde podemos dar por controladas las dos direcciones en las que puede aparecer la relación paralelo entre libros). A su vez, para los paralelos que no se habían tenido en cuenta en la precondición, se crea el hecho de paralelo_pendiente y como en la tercera parte, se guarda con ellos la marca temporal que en ejecuciones futuras de la acción leer deberá respetar, en este caso, manteniendo la distancia 0 - 1.

- **set_leido**

Acción que elimina todas las restricciones de predecesor y paralelo que tienen los libros ya leídos, dado que estas no interferiran en el plan. Esta acción solo es ejecutada para los libros que el usuario ya se ha leído previamente (libros fuera del plan) ya que los efectos de esta son comunes a la acción leer y esta nunca será necesaria para el planificador.

```
(:action set_leido
    :parameters (?l - libro)
    :precondition (and (leido ?l) (not (end)))
    :effect (and
        (forall (?p - libro)
            (when (predecesor ?l ?p)
                (not (predecesor ?l ?p))    ; Elimina las relaciones de predecesor.
            )
        )
        (forall (?p - libro)
            (when (paralelo ?l ?p) (and
                (not (paralelo ?l ?p))      ; Elimina las relaciones de paralelo
                (not (paralelo ?p ?l))      ; en ambos sentidos de la relacion
            ))
        )
    )
)
```

Necesitamos esta acción para que los libros leídos previamente leídos no interfieran en la planificación.

Si estos eran predecesor de algún libro, o alguien les precede a ellos, al estar leídos, ya cumplen con todas las condiciones. Del mismo modo ocurre con los paralelos. Si estos son paralelos a algún libro añadido al plan, simplemente ya estan leidos, y no es necesario su relectura. Por lo tanto eliminar esta relacion solo elimina elementos del dominio que facilitaran la búsqueda y simplificarán ligeramente la acción principal de leer un libro.

Remarcamos también que si hemos leído previamente un libro, y este tiene otros paralelos, no vamos a añadir al plan estos últimos. Desde nuestro punto de vista no tiene sentido y además carecemos del mes de lectura de este y, por lo tanto, no podemos asignarlos a distancia 0 o 1 del libro ya leído.

- **end**

Acción que se ejecuta al final de todo plan para comprobar que no queden libros paralelos pendientes de leer.

```
(:action end
    :parameters ()
    :precondition (not (exists (?p - libro ?m - mes)(paralelos_pendientes ?p ?m)))
    :effect (end)
)
```

Dado que nosotros hemos escogido representar la relación de paralelo en una dirección (a pesar de ser una relación simétrica) para facilitar el trabajo a la acción leer del planificador, puede pasar que, dependiendo de como están definidas estas relaciones, el planificador tenga en el sistema un conjunto de libros paralelos pendientes de asignar pero ya cumpla con todos los requisitos del “goal”.

Para controlar esta situación, decidimos añadir al “goal” el predicado (end). De este modo, el programa se ve obligado a pasar por esta última acción que solo le da el visto bueno si no se ha dejado libros paralelos sin planificar.

MODELIZACIÓN DEL PROBLEMA

Nivel Básico + Extension 1

Objetos

En este caso disponemos de dos clases de objetos diferentes, la clase libro y la clase mes.

```
(:objects a1 a2 a3 a4 b1 b2 c1 d1 d2 d3 e1 e2 f1 f2 f3 f4 f5 g1 g2 g3 h1  - libro
          diciembre noviembre octubre setiembre agosto julio junio mayo abril marzo febrero enero - mes
)
```

La clase libro representa los diferentes libros que han sido recomendados y entre ellos hay una selección que ya están leídos o son predecesores de otros.

La clase mes se utiliza para representar los doce meses del año donde se van a poder leer los libros que han sido recomendados.

Estado inicial

Nuestro estado inicial siempre tiene una parte estática, esta es la que expresa la anterioridad de los meses entre ellos, es decir, enero es anterior a todos sus posteriores y de esta forma para cada mes del año.

```
(anterior agosto setiembre)
(anterior agosto octubre)
(anterior agosto noviembre)
(anterior agosto diciembre)
(anterior setiembre octubre)
(anterior setiembre noviembre)
(anterior setiembre diciembre)
(anterior octubre noviembre)
(anterior octubre diciembre)
(anterior noviembre diciembre)
```

A continuación se expresan las relaciones que indican que libros son predecesores de otros.

```
(predecesor a1 a2)
(predecesor a2 a3)
(predecesor a3 a4)
(predecesor b1 b2)
(predecesor d1 d2)
(predecesor d2 d3)
(predecesor e1 e2)
(predecesor f1 f2)
(predecesor f2 f3)
(predecesor f3 f4)
(predecesor f4 f5)
(predecesor g1 g2)
(predecesor g2 g3)

(predecesor a4 f3)
(predecesor d2 a2)
(predecesor a1 e2)
(predecesor e2 d1)
```

Finalmente se especifican qué libros ya están leídos con anterioridad.

```
(leido f1)
(leido a1)
(leido a2)
(leido e2)
```

Objetivo

En esta sección del problema se indican qué libros deben estar marcados como leídos para que se dé por completada la planificación independientemente de los libros que por obligación de deben leer.

```
(:goal
    (and
        (leido f5)
        (leido g2)
    )
)
```

```
(:goal
    (forall (?l - libro) (leido ?l))
)
```

Extension 2

Objetos

En esta segunda extensión el apartado de objetos no ha sufrido ningún cambio ya que los elementos que representan el problema siguen siendo los mismos.

Estado inicial

En este caso sí que se sufren modificaciones respecto a la extensión anterior, al haber sido añadida la distancia 0-1 entre los meses, esta debe ser especificado en el estado inicial.

```
(dist01 enero enero)
(dist01 enero febrero)
(dist01 febrero enero)
(dist01 febrero febrero)
(dist01 febrero marzo)
(dist01 marzo febrero)
(dist01 marzo marzo)
(dist01 marzo abril)
```

Las anteriores descripciones del estado inicial siguen siendo las mismas que las del estado anterior.

Objetivo

Lo único que se ha añadido es el predicado (end) para que el programa se vea forzado a terminar cuando se hayan procesado todos los libros que se deben tratar.

```
(:goal
    (and
        (leido f5)
        (leido c1)
        (end)
    )
)
```

DESARROLLO DE LOS MODELOS

Nuestro desarrollo ha sido incremental. Empezamos por el nivel básico, y una vez obtuvimos un dominio funcional para este nivel, pasamos al siguiente. Como hemos mencionado anteriormente en el documento, directamente obtuvimos un código funcional para la primera extensión.

Para la segunda extensión, pudimos mantener todo el código que teníamos. No necesitamos borrar nada, simplemente añadir las líneas referentes al segundo apartado, siguiendo un razonamiento similar aunque con una vuelta de tuerca extra debido a la simetría de la relación entre paralelos.

Los generadores de juegos de prueba los creamos paralelamente al desarrollo del modelo (nos encargamos integrantes del grupo diferentes).

JUEGOS DE PRUEBA

A continuación se muestra la ejecución de varios juegos de prueba, ejecutados con la extensión 2. Mostramos su input, su output y un breve análisis de cada uno.

Juego de prueba 1

Input

```
(define (problem Book-Planner)
        (:domain books)
        (:objects a b c d e f g h i j k l m n o p q r s t u v w x y z - libro
          enero febrero marzo abril mayo junio julio agosto setiembre octubre
          noviembre diciembre - mes
   )
        (:init

        (anterior enero febrero)(anterior enero marzo).....
        (dist01 enero enero)(dist01 enero febrero).....

        (leido a)(leido c)(leido d)(leido e)(leido f)(leido j)(leido k)(leido m)(leido n)
        (leido o)(leido p)(leido t)(leido v)(leido x)(leido z)
        )
        (:goal
                (and
                        (leido b)(leido g)(leido h)(leido i)(leido l)(leido q)(leido r)
                        (leido s)(leido u)(leido w)(leido y)
                        (end)
)       )       )
```

Output

```
step 0: LEER B DICIEMBRE
      1: LEER G DICIEMBRE
      2: LEER H DICIEMBRE
      3: LEER I DICIEMBRE
      4: LEER L DICIEMBRE
      5: LEER Q DICIEMBRE
      6: LEER R DICIEMBRE
      7: LEER S DICIEMBRE
```

8: LEER U DICIEMBRE
9: LEER W DICIEMBRE
10: LEER Y DICIEMBRE
11: END

0.01 seconds total time

Análisis

Este problema tiene 26 libros, y un subconjunto de estos ya han sido leídos en el pasado y el resto son los que se quieren leer. Con este problema se genera un plan muy sencillo. Ya que no hay dependencias entre los libros, se pueden leer todos en el mismo mes.

Juego de prueba 2

Input

```
(define (problem Book-Planner)
        (:domain books)
        (:objects a b c d e f g h - libro
          enero febrero marzo abril mayo junio julio agosto setiembre octubre
          noviembre diciembre - mes
        )
        (:init

                (anterior enero febrero)(anterior enero marzo).....
                (dist01 enero enero)(dist01 enero febrero).....

                (predecesor a b)(predecesor b c)(predecesor c d)
                (predecesor e f)(predecesor f g)(predecesor g h)
                (paralelo d e)

        )
        (:goal
                (and
                        (leido h)
                        (end)
)       )       )
```

Output

step 0: LEER A ENERO
1: LEER B FEBRERO
2: LEER C MARZO
3: LEER D ABRIL
4: LEER E MARZO
5: LEER F ABRIL
6: LEER G MAYO
7: LEER H DICIEMBRE
8: END

0.01 seconds total time

Análisis

Este problema tiene dos “sagas” de libros. Una formada por ‘a, b, c, d’ y la otra por ‘e, f, g, h’. Además el libro ‘d’ es paralelo al ‘e’. Ya que el objetivo es leer el libro ‘h’, se tendrán que leer todos sus predecesores en meses anteriores. Al estar ‘e’ incluido en la cadena de predecesores, y tener a ‘d’ como paralelo, se tendrá que leer ‘d’ en un mes a distancia 1, y a su vez leer toda su saga de predecesores, ‘a, b, c’. El resultado muestra como los meses están bien escogidos.

Juego de prueba 3

Input

```
(define (problem Book-Planner)
        (:domain books)
        (:objects a b c d e f g h i j k l - libro
          enero febrero marzo abril mayo junio julio agosto setiembre octubre
          noviembre diciembre - mes
         )
        (:init

        (anterior enero febrero)(anterior enero marzo).....
        (dist01 enero enero)(dist01 enero febrero).....

        (predecesor a e)(predecesor b d)(predecesor c k)(predecesor d k)
        (predecesor f i)(predecesor g j)
        (paralelo a g)(paralelo b g)(paralelo c l)(paralelo d f)(paralelo e k)(paralelo h j)
        )
```

```
        (:goal
                (and
                        (leido a)(leido b)(leido c)(leido d)
                        (leido e)(leido f)(leido g)(leido h)
                        (leido i)(leido j)(leido k)(leido l)
                        (end)
)       )       )
```

Output

```
step 0: LEER H ENERO
       1: LEER C ENERO
       2: LEER L FEBRERO
       3: LEER B ENERO
       4: LEER D FEBRERO
       5: LEER F ENERO
       6: LEER I DICIEMBRE
       7: LEER A ENERO
       8: LEER E FEBRERO
       9: LEER K MARZO
      10: LEER G ENERO
      11: LEER J FEBRERO
      12: END

0.02 seconds total time
```

Análisis

Este problema es más complejo que los anteriores, y pretende poner a prueba al planificador. En este caso, tenemos 12 libros, 6 relaciones de paralelos y 6 de predecesores. El fin es ver cómo organiza los libros cumpliendo con todos los requisitos. La salida muestra cómo efectivamente los libros paralelos son leídos con una diferencia de un mes entre ellos, y todos los predecesores son leídos en meses anteriores a los libros que preceden.

Juego de prueba 4

Input

```
(define (problem Book-Planner)
        (:domain books)
```

```
        (:objects a b c d e f g h i j k l m n o p q r s t u v w x y z - libro
          enero febrero marzo abril mayo junio julio agosto setiembre octubre
          noviembre diciembre - mes
    )
        (:init

        (anterior enero febrero)(anterior enero marzo).....
        (dist01 enero enero)(dist01 enero febrero).....

        (predecesor a i)(predecesor b v)(predecesor c f)(predecesor d x)
        (predecesor e w)(predecesor g h)(predecesor j n)(predecesor k u)
        (predecesor p v)(predecesor q y)(predecesor t x)(predecesor u y)
        (predecesor v y)

        (paralelo b s)(paralelo c h)(paralelo d n)(paralelo e j)(paralelo h j)
        (paralelo i q)(paralelo j x)(paralelo k u)(paralelo l q)(paralelo m o)(paralelo o x)

        )
        (:goal
                (and
                        (leido a)(leido b)(leido c)(leido d)(leido e)(leido f)(leido g)
                        (leido h)(leido i)(leido j)(leido k)(leido l)(leido m)(leido n)
                        (leido o)(leido p)(leido q)(leido r)(leido s)(leido t)(leido u)
                        (leido v)(leido w)(leido x)(leido y)(leido z)
                        (end)
)       )       )
```

Output

```
step 0: LEER Z DICIEMBRE
       1: LEER T ENERO
       2: LEER R DICIEMBRE
       3: LEER P ENERO
       4: LEER M ENERO
       5: LEER O ENERO
       6: LEER L ENERO
       7: LEER K ENERO
       8: LEER U FEBRERO
       9: LEER G ENERO
      10: LEER E ENERO
      11: LEER W DICIEMBRE
      12: LEER D ENERO
```

13: LEER C ENERO
14: LEER F DICIEMBRE
15: LEER H FEBRERO
16: LEER J ENERO
17: LEER N FEBRERO
18: LEER X FEBRERO
19: LEER B ENERO
20: LEER V FEBRERO
21: LEER S FEBRERO
22: LEER A ENERO
23: LEER I FEBRERO
24: LEER Q ENERO
25: LEER Y DICIEMBRE
26: END

0.09 seconds total time

Análisis

Este problema tiene 26 libros, y ha sido creado con un generador de problemas aleatorios, con la máxima probabilidad de que un libro sea predecesor de otro y lo mismo con ser predecesor de otro libro. Tenemos muchos libros y muchas relaciones entre ellos, pero el plan generado es correcto. Solamente se utilizan 3 meses para repartir todos los libros, suficiente para cumplir con los requisitos de lectura. El tiempo de ejecución es relativamente bajo para la complejidad del problema.

Generador de juegos de prueba

Como parte del proyecto, hemos programado un generador de juegos de prueba en el lenguaje Java. Este programa permite generar de forma aleatoria ficheros “problema.pddl” a partir de unos parámetros que el usuario introduce por terminal. Las relaciones entre los libros de los problemas generados se pueden representar con un grafo dirigido acíclico. Con lo cual nunca se producirán ciclos que hagan que el planificador no pueda generar un plan. El funcionamiento del programa es el siguiente.

1. Introducir el número de libros a generar, entre 1 y 26, los cuales serán nombrados con letras de la ‘a’ a la ‘z’.

2. Introducir la probabilidad de que un libro sea predecesor de otro , entre 0 y 1. Esto generará los hechos (predecesor x y), donde X e Y son libros.

3. Introducir la probabilidad de que un libro sea paralelo a otro otro , entre 0 y 1. Esto generará los hechos (paralelo x y), donde X e Y son libros.

4. Introducir los libros que se ha leído. Los libros deben estar entre los creados anteriormente, si no el programa generará hechos con libros inexistentes. Se pueden seleccionar todos a la vez escribiendo ‘all’. Esto generará las hechos de (leido x) en la parte “init” del problema, donde x es un libro.

5. Introducir los libros que se se quiere leer (goal). De igual manera, los libros deben estar entre los creados anteriormente, si no el programa generará hechos con libros inexistentes. Se pueden seleccionar todos a la vez escribiendo ‘all’. Esto generará los hechos de (leido x) en la parte “goal” del problema, donde x es un libro.

6. El programa genera automáticamente un problema con las características indicadas.

Análisis del tiempo de ejecución

Para analizar el rendimiento del planificador, hemos generado un conjunto de problemas y hemos hecho gráficas que muestran la función de crecimiento del tiempo en cada caso (utilizando el dominio de la extensión 2).

Hemos experimentado con problemas de tamaños 5, 10, 15, 20 y 25 libros. Para cada tamaño, hemos hecho tres ejecuciones.

1. Genera hechos de libros paralelos con probabilidad 0, y genera hechos de libros predecesores con probabilidad ‘p’ (0...0,9).

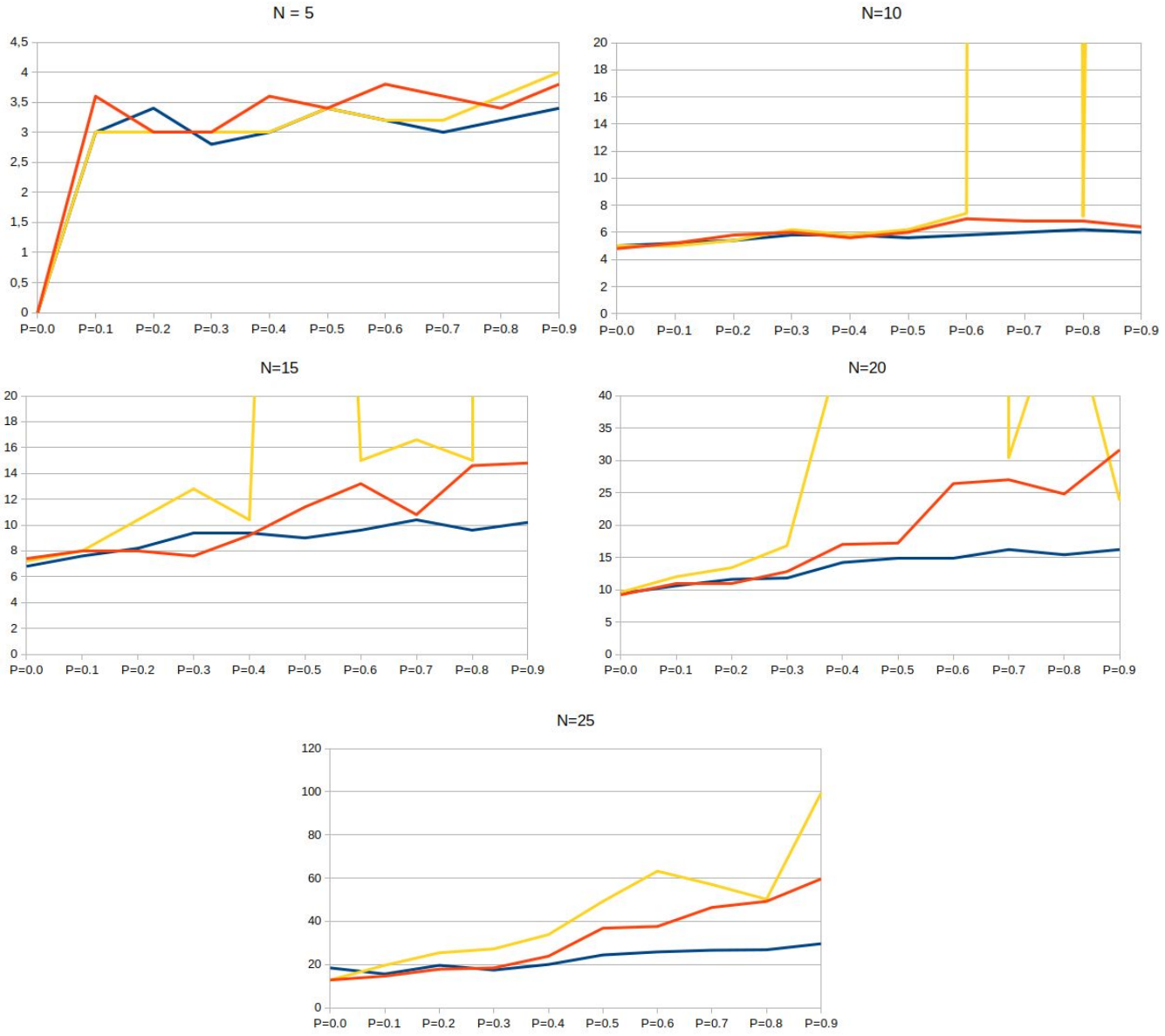
2. Genera hechos de libros predecesores con probabilidad 0, y genera hechos de libros paralelos con probabilidad ‘p’ (0...0,9).

3. Genera hechos de libros predecesores con probabilidad ‘p’ (0...0,9), y genera hechos de libros paralelos con la misma probabilidad ‘p’.

A continuación se muestran las gráficas, donde el eje X representa la probabilidad p y el eje Y representa el tiempo de ejecución en milisegundos. Los tiempos han sido

calculados haciendo la media de 5 ejecuciones con 5 problemas diferentes pero con los mismos parámetros del generador de juegos de prueba.

Rojo: Caso 1 Azul: Caso 2 Amarillo: Caso 3



Los casos 1 y 2 se comportan de forma muy parecida, ya que la complejidad del problema es similar. Además los planes se calculan de manera relativamente rápida.

El caso 3 es el que tiene más complejidad. En este caso, el planificador debe generar un plan que satisfaga los requisitos de predecesores y paralelos, lo cual no es tan sencillo. Si se desea leer un libro hay que leer todos sus predecesores, y si algún predecesor tiene libros paralelos, también se deben leer estos y sus

predecesores, y así sucesivamente. Esto da lugar a casos en los que el problema tarda ya no unos milisegundos sino varios segundos.

En las gráficas anteriores para poder hacerlas comprensibles no se alcanza a ver el tiempo de algunas ejecuciones del caso 3, pero estas han alcanzado tiempos de 1 hasta 7 segundos. Estas situaciones son muy aleatorias, ya que depende de problemas concretos y sus relaciones entre los libros, más que del número de libros. Puede darse el caso en un problema con pocos libros que debido a las relaciones con otros libros el planificador tarde varios segundos en obtener un plan.

Experimentando también hemos obtenido problemas en los que el planificador ha estado procesando un plan durante muchos minutos, sin llegar a ninguna solución en concreto. Esto es debido a que hay problemas en los que el planificador no es capaz de generar un plan ya que debido al solapamiento de libros paralelos y predecesores no hay meses suficientes para leer todos los libros, porque los predecesores tienen como condición necesaria que han de ser leídos antes que el los libros de los que son predecesores.

Finalmente, cabe mencionar que en estos experimentos no hemos tenido en cuenta la posibilidad de que el usuario haya leído algún libro de antemano, pero esto realmente sólo facilita las cosas al planificador, ya que si esto pasara, el plan no incluiría ni esos libros ni sus predecesores.