Práctica de planificación

Alejandro Espinosa, Sara Buceta y Pol Casacuberta

Primavera 2022

Contents

1 Introducción

Esta práctica tiene como objetivo resolver mediante un planificador en PDDL una versión simplificada del problema planteado para la práctica de Sistemas Basados en el Conocimiento. En este caso, la agencia de viajes *al fin del mundo y más allá* requiere un programa básico que cree planes de viaje al que se le pueden hacer distintas extensiones.

2 Modelado del dominio

2.1 Tipos de datos

2.1.1 Nivel Básico

En el nivel Básico solamente necesitamos representar las ciudades, los vuelos que usaremos para ir de una ciudad a otra y los hoteles que habrá en cada ciudad.

- ciudad - ciudad
- hotel - hotel
- vuelo - vuelo

2.1.2 Extensiones 1-4

En todas las extensiones se nos añade la complejidad de necesitar saber los días que estaremos por ciudad, por lo que creamos un nuevo tipo para poder identificar cuáles son los días por ciudad posibles para unificar con nuestro operador.

- ciudad - ciudad
- hotel - hotel
- vuelo - vuelo
- dias_por_ciudad - dias_por_ciudad

2.2 Predicados

2.2.1 Nivel Básico y Extensiones 1-4

Los únicos predicados que vamos a necesitar en esta práctica son los siguientes:

- va_a ?x - vuelo ?y - ciudad ?z - ciudad
- esta_en ?x - hotel ?y - ciudad
- ciudad_visitada ?c - ciudad
- current_ciudad ?c - ciudad

va_a indica que un vuelo va de una ciudad 1 a una ciudad 2, nos permite identificar qué vuelo va de una ciudad a otra.

esta_en indica que un hotel se encuentra dentro de una ciudad.

ciudad_visitada nos permite saber qué ciudades ya hemos visitado, con el objetivo de no volver a visitarlas en nuestro plan.

current_ciudad nos indica en qué ciudad nos encontramos actualmente.

2.3 Flujos

2.3.1 Nivel Básico

En el nivel más básico de la práctica solamente vamos a emplear dos fluentes, el número de ciudades que hemos escogido hasta el momento y el mínimo de ciudades a escoger, cuando el número de ciudades escogidas sea mayor al mínimo llegaremos al goal.

- num_ciudades_escogidas
- min_ciudades_a_recoger

2.3.2 Extensión 1

En la extensión 1 se añaden 5 flujos nuevos, además de los fluentes del nivel básico, dado que debemos tener en cuenta el número de días que hemos recorrido hasta el momento, ya que se nos da un mínimo de días que debe de tener el recorrido, para lo que también habrá que añadir un flujo. También deberemos representar los días por ciudad con flujos, dado un objeto de tipo dias_por_ciudad deberemos poder asociar a ese objeto el número de días que representa. También debemos de representar el mínimo y máximo de días por ciudad.

- num_ciudades_escogidas
- num_dias_recorrido
- min_ciudades_a_recoger
- min_dias_recorrido
- min_dias_por_ciudad
- max_dias_por_ciudad
- dias_por_ciudad ?x - dias_por_ciudad

2.3.3 Extensión 2

En esta segunda extensión, además de los fluentes de la extensión 1, añadimos los fluentes interes_ciudad, que asocian una ciudad con el número que representa su interés. También debemos tener en cuenta cuánto interés tienen las ciudades que hemos escogido hasta el momento, ya que queremos maximizar el interés (minimizando el valor, ya que a menor valor mayor interés).

- num_ciudades_escogidas
- num_dias_recorrido
- min_ciudades_a_recoger
- min_dias_recorrido
- min_dias_por_ciudad
- max_dias_por_ciudad
- dias_por_ciudad ?x - dias_por_ciudad
- interes_ciudad ?c - ciudad
- interes_actual

2.3.4 Extensión 3

En esta tercera extensión tomamos de partida los fluentes de la extensión 1 y añadimos fluentes para representar el precio actual del plan que conforme añadamos ciudades y hoteles incrementara, por lo tanto, debemos de añadir un fluente de precio_hotel que asocie un hotel con su precio, y otro fluente precio_vuelo que asocia el vuelo con su precio. También debemos de añadir dos fluentes para representar el máximo precio del plan que no podremos sobrepasar y un mínimo de precio del plan que deberemos superar.

- num_ciudades_escogidas
- num_dias_recorrido
- min_ciudades_a_recoger
- min_dias_recorrido
- min_dias_por_ciudad

- max_dias_por_ciudad
- dias_por_ciudad ?x - dias_por_ciudad
- min_precio_plan
- max_precio_plan
- precio_plan
- precio_hotel ?h - hotel
- precio_vuelo ?v - vuelo

2.3.5 Extensión 4

En esta extensión número 4 usaremos los fluentes de la extensión 1 y añadimos los que hemos agregado en la extensión 2 y también los que hemos agregado en la extensión 3.

- num_ciudades_escogidas
- num_dias_recorrido
- min_ciudades_a_recoger
- min_dias_recorrido
- min_dias_por_ciudad
- max_dias_por_ciudad
- dias_por_ciudad ?x - dias_por_ciudad
- min_precio_plan
- max_precio_plan
- precio_plan
- precio_hotel ?h - hotel
- precio_vuelo ?v - vuelo
- interes_ciudad ?c - ciudad
- interes_actual

2.4 Operadores

2.4.1 Nivel Básico

Para construir nuestra solución al problema no requerimos más que un operador, ya que el problema solo nos pide escoger un recorrido entre ciudades. Nuestro operador anadir_ciudad, en su versión del nivel básico, toma como parámetros una ciudad donde se encuentra ahora mismo el usuario, otra ciudad a donde el usuario va a viajar, el vuelo que va lo va a llevar desde la ciudad 1 a la ciudad 2 y el hotel donde se va a hospedar. Para que el operador se pueda aplicar se debe añadir una ciudad fantasma y vuelos fantasmas (usaremos solo un objeto de vuelo que pueda viajar a cualquier ciudad con ese objeto único) desde esa ciudad a todas las ciudades, ya que necesitamos establecer una current_ciudad y como la ciudad origen del cliente es desconocida empleamos esta ciudad fantasma. Nuestro current_ciudad al inicio será esta ciudad fantasma, para poder aplicar el operador siempre que escojamos la primera ciudad.

Para poder aplicar el operador, el hotel escogido debe de pertenecer a la ciudad a la que vamos. La ciudad no debe de pertenecer ya al recorrido de ciudades que llevamos hasta el momento y el vuelo escogido debe de llevarnos de la ciudad 1 a la ciudad 2.

El efecto del operador será marcar la ciudad 2 como visitada, y la ciudad_actual como ciudad 2, quitará ciudad 1 de ciudad_actual. Por último, incrementará el fluente de número de ciudades escogidas en 1.

2.4.2 Extensión 1

Este operador extenderá la funcionalidad del operador básico añadiendo el uso de los siguientes fluentes:

- min_ciudades_a_recoger
- min_dias_recorrido
- min_dias_por_ciudad
- max_dias_por_ciudad
- dias_por_ciudad ?x - dias_por_ciudad

Ya que en esta extensión se nos pide el mínimo y máximo de días que hay que estar en las ciudades y el mínimo número de ciudades a visitar. El operador recibirá como parámetro dias_por_ciudad que nos indicará el número de días que se visitará esa ciudad, en la precondición comprobamos que el número de días por ciudad es mayor a min_dias_por_ciudad y que es menor que max_dias_por_ciudad.

En el efecto incrementamos el num_dias_recorrido por el valor de dias_por_ciudad.

2.4.3 Extensión 2

Este operador extiende la funcionalidad del operador de la Extensión 1 y añade el uso de los fluentes:

- interes_ciudad ?c - ciudad
- interes_actual

El operador toma los mismos parámetros que el de la extensión 1 y tiene las mismas precondiciones, sin embargo, en los efectos incrementamos el interés total del viaje "interes_actual" sumando el interés de la ciudad añadida. Hacemos esto ya que se nos pide maximizar el interés de las ciudades, por lo tanto, debemos mantener la cuenta del interés acumulado.

2.4.4 Extensión 3

Este operador nuevamente extiende la funcionalidad del operador de la Extensión 1 y añade el uso de los siguientes fluentes:

- min_precio_plan
- max_precio_plan
- precio_plan
- precio_hotel ?h - hotel
- precio_vuelo ?v - vuelo

El operador toma los mismos parámetros que el de la extensión 1, pero añadimos una precondición y es que el precio_vuelo (el precio del vuelo hasta este instante) más la suma del vuelo más el precio del hotel multiplicado por los días que estás en la ciudad debe de ser menor al precio máximo del plan. En los efectos, añadimos el incremento del precio del plan, sumándole el precio del vuelo y el del hotel por los días de estancia.

2.4.5 Extensión 4

Este operador combina la funcionalidad de la Extensión 2 y de la extensión 3. El operador toma como parámetros los mismos que el de la extensión 1, las precondiciones son las de la extensión 1 añadiendo las agregadas en la extensión 2 y las de la extensión 3 y los efectos también son los de la extensión 1 añadiendo los agregados en la extensión 2 y en la extensión 3. No hay más cambios relevantes.

3 Modelado de los problemas

3.1 Objetos

3.1.1 Nivel Básico

En el nivel básico modelamos las ciudades, los vuelos que van a llevarnos de una ciudad a otra y los hoteles que van a pertenecer a alguna ciudad. Necesitamos añadir una ciudad fantasma y vuelos desde la ciudad fantasma a todas las demás ciudades, ya que no conocemos la ciudad de la que proviene el usuario, así que tenemos que asumir que el recorrido puede empezar desde cualquier ciudad.

- cg1 c1 ... cn - ciudad
- vg1 v1 ... vm - vuelo
- h1 ... hn - hotel

3.1.2 Extensión 1-4

En las extensiones de la primera a la cuarta añadimos los días que estaremos por ciudad, ya que necesitamos poder unificar con todos los días por ciudad posibles.

- cg1 c1 ... cn - ciudad
- vg1 v1 ... vm - vuelo
- h1 ... hn - hotel
- dias1 ... diasx - dias_por_ciudad

3.2 Estado inicial

3.2.1 Nivel Básico

En el nivel básico el estado inicial consiste en inicializar el número de ciudades escogidas hasta el momento a 0, el mínimo de ciudades a recoger según el valor que nos interese, en este caso a modo de ejemplo lo iniciamos a 3.

También debemos escribir la ciudad actual, colocaremos la ciudad fantasma en este predicado y como ya hemos dicho previamente, creamos el vuelo fantasma de forma que pueda ir desde la ciudad fantasma a cualquier ciudad.

Luego colocaremos los vuelos que queramos y seguidamente colocaremos a que ciudad pertenecen los hoteles.

```
(= (num_ciudades_escogidas) 0)
(= (min_ciudades_a_recoger) 3)
(current_ciudad cg1)
(ciudad_visitada cg1)
(va_a vg1 cg1 c1)
(va_a vg1 cg1 c2)
...
(va_a vg1 cg1 cn)
(va_a v1 c1 c2)
...
(esta_en h1 c1)
...
```

3.2.2 Extensión 1

En esta extensión tomaremos como base el nivel básico y añadiremos las siguientes inicializaciones:

```
(= (min_dias_por_ciudad) 2)
(= (max_dias_por_ciudad) 4)
(= (num_dias_recorrido) 0)
(= (min_dias_recorrido) 10)
```

```
(= (dias_por_ciudad dias2) 1)
(= (dias_por_ciudad dias2) 2)
(= (dias_por_ciudad dias3) 3)
(= (dias_por_ciudad dias4) 4)
```

Hemos de añadir un mínimo y un máximo de días por ciudad, sea el que sea, ya que dependerá de eso si hay solución o no. También debemos inicializar el número de días que lleva el recorrido a 0 y fijar un mínimo de días por ciudad. También deberemos añadir el valor de los días por ciudad que se van a poder unificar, deberían estar entre el máximo y el mínimo, ya que si no no se van a usar y deberían estar todos los enteros entre el mínimo y el máximo, ya que, sino algún valor entre medio del rango, no se va a poder usar.

3.2.3 Extensión 2

En esta segunda extensión tomamos como base las inicializaciones de la extensión 1. Debemos de inicializar el interés actual que debe de ser 0, conforme añadamos ciudades, el interés de esas ciudades se añadirá a este fluente. También debemos inicializar los valores de los intereses de las distintas ciudades, el valor del interés de la ciudad fantasma no importa, ya que no se va a usar.

```
(= (interes_actual) 0)
(= (interes_ciudad c1) 1)
(= (interes_ciudad c2) 2)
...
(= (interes_ciudad cn) 3)
(= (interes_ciudad cg1) 3)
```

3.2.4 Extensión 3

En esta tercera extensión tomaremos como base las inicializaciones de la extensión 1 y añadiremos las siguientes:

```
(= (min_precio_plan) 300)
(= (max_precio_plan) 4000)
(= (precio_plan) 0)

(= (precio_hotel h1) 300)
(= (precio_hotel h2) 150)
(= (precio_hotel h3) 700)
...
(= (precio_hotel hn) 700)

(= (precio_vuelo vg1) 0)
(= (precio_vuelo v1) 100)
(= (precio_vuelo v2) 150)
(= (precio_vuelo v3) 150)
...
(= (precio_vuelo vm) 150)
```

Debemos inicializar el precio mínimo y máximo que debe de tener el plan, sea el que sea, aunque si somos demasiado restrictivos no habrá solución posible. Hay que inicializar el precio actual del plan a 0 y por último hay que inicializar los precios de los hoteles y el de los vuelos, asociándolos con sus respectivos precios.

3.2.5 Extensión 4

En la extensión 4 partimos de las inicializaciones de la extensión básica, y agregamos las de la extensión 1, 2 y 3.

3.3 Estado final

3.3.1 Nivel Básico

En el nivel básico, para que un plan sea válido, solamente tiene que cumplir que el número de ciudades escogidas sea mayor al mínimo.

- (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))

3.3.2 Extensión 1-2

En las extensiones 1 y 2 el objetivo es encontrar una planificación donde el número de ciudades escogidas sea mayor al mínimo de ciudades a escoger y además el número de días del recorrido debe de ser mayor al número de días mínimo.

- (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
- (<= (min_dias_recorrido) (num_dias_recorrido))

3.3.3 Extensión 3-4

En las extensiones 3 y 4 además de las restricciones que encontramos en las extensiones 1 y 2, se les añaden dos restricciones más y es que el precio del plan debe de estar entre el máximo y el mínimo precio del plan.

- (<= (min_precio_plan) (precio_plan))
- (>= (max_precio_plan) (precio_plan))
- (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
- (<= (min_dias_recorrido) (num_dias_recorrido))

3.4 Métricas

En el nivel básico y en la extensión 1 no se utiliza ninguna métrica a minimizar ni maximizar.

3.4.1 Extensión 2

En esta extensión buscamos minimizar el interés total, ya que las ciudades con mayor interés tienen el valor más bajo.

- minimize (interes_actual)

3.4.2 Extensión 3

En esta extensión se nos pide minimizar el precio del plan con el siguiente fluente.

- minimize (precio_plan)

3.4.3 Extensión 4

En esta última extensión utilizamos la siguiente función para minimizar ambos parámetros, para que "interes_actual" sea relevante le añadimos un multiplicador, que después de hacer pruebas nos ha parecido adecuado.

- minimize (+ (precio_plan) (* (interes_actual) 400))

4 Desarrollo de las distintas extensiones

El nivel básico se ha desarrollado por iteraciones, en una primera versión se usaban más predicados como por ejemplo alojamientos_escogidos o vuelos_escogidos, pero nos dimos cuenta de que no eran necesarios estos predicados, ya que no puedes visitar más de una vez la misma ciudad y asumimos no puedes estar en dos hoteles en una misma ciudad, por lo que no es necesario llevar la cuenta de los hoteles y los vuelos que has cogido. Quitando esos predicados reducimos la complejidad.

En la extensión 1 pensamos en tratar los días por ciudad con predicados pero, al averiguar como funcionaban los fluentes, nos dimos cuenta de que sería una tarea mucho más fácil haciendo uso de ellos.

En la extensión 3 pensamos en añadir predicados nuevos para los precios de los alojamientos y los de los hoteles, pero realmente no son predicados nuevos, tienen que ser fluentes, ya que deben de contener un valor. Además, pensamos en añadir un tipo nuevo para estos fluentes, pero no es necesario ya que lo que queremos es asociar a cada alojamiento y a cada vuelo un precio y lo podemos hacer con un fluente precio_vuelo que tome un vuelo y un precio_alojamiento que tome un alojamiento.

5 Juegos de prueba

En este apartado de la práctica comprobaremos el correcto funcionamiento de todas las funcionalidades del código, analizaremos para cada extensión 3 juegos de prueba distintos, cabe destacar que cada uno de estos juegos de prueba han sido generados por un script de Python. Cada extensión tiene su propio script, y es muy fácil de usar, simplemente hay que responder unas pocas preguntar y copiar la salida en un archivo PDDL (problema). Este es el punto extra de la práctica, los scripts se pueden encontrar en la carpeta de cada extensión.

5.1 Nivel básico

En el nivel básico buscaremos ver que se seleccionan correctamente las ciudades, hoteles y vuelos. Además comprobaremos que se respeta el número mínimo de ciudades a visitar.

5.1.1 Prueba 1

Explicación juego de prueba
En este caso usaremos un juego de prueba de tamaño pequeño, con 4 ciudades, 4 aviones y 4 hoteles, además querremos estar mínimo 4 días por ciudad.

Input (problema)

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1  c1  c2  c3  c4   - ciudad
        vg1  v1  v2  v3  v4   - vuelo
        h1  h2  h3  h4  - hotel
    )
    (:init
        (= (num_ciudades_escogidas) 0)
        (= (min_ciudades_a_recoger) 4 )
        (current_ciudad cg1)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
        (va_a vg1 cg1 c2)
        (va_a vg1 cg1 c3)
        (va_a vg1 cg1 c4)
        (va_a v1 c2 c1 )
        (va_a v2 c1 c3 )
        (va_a v3 c3 c4 )
        (va_a v4 c4 c2 )
        (esta_en h1 c4 )
        (esta_en h2 c2 )
        (esta_en h3 c1 )
        (esta_en h4 c3 )

    )

    (:goal (and
    (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
    ))
)
```

Output obtenido

```
advancing to distance:    4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C3 VG1 H4
        1: ANADIR_CIUDAD C3 C4 V3 H1
        2: ANADIR_CIUDAD C4 C2 V4 H2
        3: ANADIR_CIUDAD C2 C1 V1 H3


time spent:    0.00 seconds instantiating 8 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 14 facts and 8 actions
               0.00 seconds creating final representation with 13 relevant facts, 1 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 8 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Como podemos observar comparando el input y el output obtenido, el planificador básico funciona perfectamente para este juego de pruebas, asignando a las ciudades hoteles que pertenecen a estas y asignando vuelos entre ciudades correctamente (el vuelo parte y va a las ciudades adecuadas). Además, podemos ver como se respeta el mínimo número de ciudades a visitar(4).

5.1.2 Prueba 2

Explicación juego de prueba
En este caso usaremos un juego de prueba de tamaño grande, el objetivo es comprobar lo mismo que en el juego de pruebas 1, pero con un mínimo de ciudades a visitar menor al número de ciudades, además del correcto funcionamiento para muchas instancias.

Usaremos 20 ciudades, 30 vuelos, 25 hoteles y mínimo visitaremos 15 ciudades.

Input (problema)

```
(define (problem agencia_viaje)
   (:domain agencia_viaje)
   (:objects
      cg1 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18 c19 c20
          - ciudad
      vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20
          v21 v22 v23 v24 v25 v26 v27 v28 v29 v30  - vuelo
      h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21
          h22 h23 h24 h25 - hotel
   )
   (:init
      (= (num_ciudades_escogidas) 0)
      (= (min_ciudades_a_recoger) 15 )
      (current_ciudad cg1)
      (ciudad_visitada cg1)
      (va_a vg1 cg1 c1)
      (va_a vg1 cg1 c2)
      (va_a vg1 cg1 c3)
      (va_a vg1 cg1 c4)
      (va_a vg1 cg1 c5)
```

```
(va_a vg1 cg1 c6)
(va_a vg1 cg1 c7)
(va_a vg1 cg1 c8)
(va_a vg1 cg1 c9)
(va_a vg1 cg1 c10)
(va_a vg1 cg1 c11)
(va_a vg1 cg1 c12)
(va_a vg1 cg1 c13)
(va_a vg1 cg1 c14)
(va_a vg1 cg1 c15)
(va_a vg1 cg1 c16)
(va_a vg1 cg1 c17)
(va_a vg1 cg1 c18)
(va_a vg1 cg1 c19)
(va_a vg1 cg1 c20)
(va_a v1 c9 c20 )
(va_a v2 c20 c13 )
(va_a v3 c13 c11 )
(va_a v4 c11 c14 )
(va_a v5 c14 c5 )
(va_a v6 c5 c17 )
(va_a v7 c17 c6 )
(va_a v8 c6 c19 )
(va_a v9 c19 c10 )
(va_a v10 c10 c16 )
(va_a v11 c16 c4 )
(va_a v12 c4 c1 )
(va_a v13 c1 c12 )
(va_a v14 c12 c18 )
(va_a v15 c18 c7 )
(va_a v16 c7 c8 )
(va_a v17 c8 c15 )
(va_a v18 c15 c2 )
(va_a v19 c2 c3 )
(va_a v20 c3 c9 )
(va_a v21 c8 c4 )
(va_a v22 c4 c18 )
(va_a v23 c18 c5 )
(va_a v24 c5 c14 )
(va_a v25 c14 c11 )
(va_a v26 c11 c10 )
(va_a v27 c10 c19 )
(va_a v28 c19 c20 )
(va_a v29 c20 c3 )
(va_a v30 c3 c9 )
(esta_en h1 c1 )
(esta_en h2 c8 )
(esta_en h3 c4 )
(esta_en h4 c18 )
(esta_en h5 c5 )
(esta_en h6 c14 )
(esta_en h7 c11 )
(esta_en h8 c10 )
(esta_en h9 c19 )
(esta_en h10 c20 )
(esta_en h11 c3 )
(esta_en h12 c9 )
(esta_en h13 c2 )
(esta_en h14 c16 )
(esta_en h15 c13 )
(esta_en h16 c15 )
(esta_en h17 c12 )
(esta_en h18 c17 )
```

```
        (esta_en h19 c7 )
        (esta_en h20 c6 )
        (esta_en h21 c1 )
        (esta_en h22 c8 )
        (esta_en h23 c4 )
        (esta_en h24 c18 )
        (esta_en h25 c5 )

    )

    (:goal (and
    (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
    ))
)
```

Output obtenido

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

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C6 VG1 H20
        1: ANADIR_CIUDAD C6 C19 V8 H9
        2: ANADIR_CIUDAD C19 C10 V9 H8
        3: ANADIR_CIUDAD C10 C16 V10 H14
        4: ANADIR_CIUDAD C16 C4 V11 H23
        5: ANADIR_CIUDAD C4 C1 V12 H1
        6: ANADIR_CIUDAD C1 C12 V13 H17
        7: ANADIR_CIUDAD C12 C18 V14 H24
        8: ANADIR_CIUDAD C18 C7 V15 H19
        9: ANADIR_CIUDAD C7 C8 V16 H22
       10: ANADIR_CIUDAD C8 C15 V17 H16
       11: ANADIR_CIUDAD C15 C2 V18 H13
       12: ANADIR_CIUDAD C2 C3 V19 H11
       13: ANADIR_CIUDAD C3 C9 V20 H12
       14: ANADIR_CIUDAD C9 C20 V1 H10


time spent:    0.00 seconds instantiating 63 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 62 facts and 63 actions
               0.00 seconds creating final representation with 61 relevant facts, 1 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
```

```
               0.00 seconds searching, evaluating 38 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Al igual que en el caso de prueba número 1, el funcionamiento de nuestro planificador es correcto, además podemos observar como cumple la restricción del mínimo de ciudades a visitar, ya que se visitan justamente 15 ciudades. Además, el planificador es rápido incluso con un caso de prueba así de grande, como queríamos comprobar.

5.1.3 Prueba 3

Explicación juego de prueba
Para el tercer juego de pruebas vamos a comprobar que el planificador también es capaz de decir cuando no se puede hacer un viaje, para lograr esto vamos a poner un número de ciudades mínimas a visitar mayor que el número de ciudades. Usaremos 3 ciudades, 4 vuelos, 4 hoteles y un mínimo de 5 ciudades.
Input (problema)

```
(define (problem agencia_viaje)
   (:domain agencia_viaje)
   (:objects
      cg1  c1  c2  c3   - ciudad
      vg1  v1  v2  v3  v4   - vuelo
      h1  h2  h3  h4  - hotel
   )
   (:init
      (= (num_ciudades_escogidas) 0)
      (= (min_ciudades_a_recoger) 5 )
      (current_ciudad cg1)
      (ciudad_visitada cg1)
      (va_a vg1 cg1 c1)
      (va_a vg1 cg1 c2)
      (va_a vg1 cg1 c3)
      (va_a v1 c1 c2 )
      (va_a v2 c2 c3 )
      (va_a v3 c3 c1 )
      (va_a v4 c3 c2 )
      (esta_en h1 c1 )
      (esta_en h2 c3 )
      (esta_en h3 c2 )
      (esta_en h4 c1 )

   )

   (:goal (and
   (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
   ))
)
```

Output obtenido

```
advancing to distance:    5
                          4
                          3

best first search space empty! problem proven unsolvable.



time spent:    0.00 seconds instantiating 9 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 11 facts and 9 actions
```

```
0.00 seconds creating final representation with 10 relevant facts, 1 relevant fluents
0.00 seconds computing LNF
0.00 seconds building connectivity graph
0.00 seconds searching, evaluating 12 states, to a max depth of 0
0.00 seconds total time
```

Análisis de los resultados Como podemos observar, el planificador nos dice que el problema no se puede resolver, como esperábamos. Con estos tres juegos de pruebas hemos comprobado el correcto funcionamiento de nuestro planificador en su versión básica.

5.2 Extensión 1: Mínimo y máximo de días por ciudad y en total

Nuestro objetivo para esta extensión es comprobar que se respetan las restricciones de mínimo y máximo de días por ciudad, además de la duración total. El resto de funcionalidades de esta versión del planificador han sido probadas en la versión básica.

5.2.1 Prueba 1

Explicación juego de prueba
Para este primer juego de pruebas usaremos un problema pequeño para comprobar el correcto funcionamiento de esta extensión, para esto usaremos 6 ciudades, mínimo de 5 ciudades a visitar, 10 vuelos, 10 hoteles, un mínimo de 2 días por ciudad, máximo de 5 días por ciudad y mínimo 6 días de recorrido.

Input (problema)

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1  c1  c2  c3  c4  c5  c6   - ciudad
        vg1  v1  v2  v3  v4  v5  v6  v7  v8  v9  v10   - vuelo
        h1  h2  h3  h4  h5  h6  h7  h8  h9  h10 - hotel
        dias2 dias3 dias4 dias5 - dias_por_ciudad
    )
    (:init
        (= (num_ciudades_escogidas) 0)
        (= (min_ciudades_a_recoger) 5 )
        (= (min_dias_por_ciudad) 2 )
        (= (max_dias_por_ciudad) 5 )
        (= (num_dias_recorrido) 0)
        (= (min_dias_recorrido) 6 )
        (= (dias_por_ciudad dias2) 2)
        (= (dias_por_ciudad dias3) 3)
        (= (dias_por_ciudad dias4) 4)
        (= (dias_por_ciudad dias5) 5)
        (current_ciudad cg1)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
        (va_a vg1 cg1 c2)
        (va_a vg1 cg1 c3)
        (va_a vg1 cg1 c4)
        (va_a vg1 cg1 c5)
        (va_a vg1 cg1 c6)
        (va_a v1 c2 c1 )
        (va_a v2 c1 c6 )
        (va_a v3 c6 c5 )
        (va_a v4 c5 c3 )
        (va_a v5 c3 c4 )
        (va_a v6 c4 c2 )
        (va_a v7 c6 c1 )
        (va_a v8 c1 c3 )
        (va_a v9 c3 c5 )
        (va_a v10 c5 c4 )
```

```
        (esta_en h1 c2 )
        (esta_en h2 c6 )
        (esta_en h3 c1 )
        (esta_en h4 c3 )
        (esta_en h5 c5 )
        (esta_en h6 c4 )
        (esta_en h7 c2 )
        (esta_en h8 c6 )
        (esta_en h9 c1 )
        (esta_en h10 c3 )

    )

    (:goal (and
    (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
    (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
)
```

Output obtenido

```
advancing to distance:    5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C4 VG1 H6 DIAS5
        1: ANADIR_CIUDAD C4 C2 V6 H7 DIAS5
        2: ANADIR_CIUDAD C2 C1 V1 H9 DIAS5
        3: ANADIR_CIUDAD C1 C6 V2 H2 DIAS5
        4: ANADIR_CIUDAD C6 C5 V3 H5 DIAS5


time spent:    0.00 seconds instantiating 104 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 20 facts and 104 actions
               0.00 seconds creating final representation with 19 relevant facts, 2 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 12 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Como podemos observar todas las restricciones se cumplen correctamente, como podemos observar estaremos un total de 25 días de viaje, lo que es mayor que nuestro mínimo. En cada ciudad estaremos 5 días, lo que también es correcto.

5.2.2 Prueba 2

Explicación juego de prueba
En este segundo juego de pruebas intentaremos hacer que un viaje no se pueda realizar debido a que no se puede llegar al mínimo de días del recorrido. Para hacer esto usaremos 5 ciudades, mínimo 4 ciudades a visitar, 10 vuelos, 10 hoteles, mínimo 1 día por ciudad, máximo 2 días por ciudad y querremos mínimo 15 días de recorrido. **Input (problema)**

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
```

```
    (:objects
        cg1  c1  c2  c3  c4  c5   - ciudad
        vg1  v1  v2  v3  v4  v5  v6  v7  v8  v9  v10   - vuelo
        h1  h2  h3  h4  h5  h6  h7  h8  h9  h10 - hotel
        dias1 dias2 - dias_por_ciudad
    )
    (:init
        (= (num_ciudades_escogidas) 0)
        (= (min_ciudades_a_recoger) 4 )
        (= (min_dias_por_ciudad) 1 )
        (= (max_dias_por_ciudad) 2 )
        (= (num_dias_recorrido) 0)
        (= (min_dias_recorrido) 15 )
        (= (dias_por_ciudad dias1) 1)
        (= (dias_por_ciudad dias2) 2)
        (current_ciudad cg1)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
        (va_a vg1 cg1 c2)
        (va_a vg1 cg1 c3)
        (va_a vg1 cg1 c4)
        (va_a vg1 cg1 c5)
        (va_a v1 c1 c3 )
        (va_a v2 c3 c2 )
        (va_a v3 c2 c5 )
        (va_a v4 c5 c4 )
        (va_a v5 c4 c1 )
        (va_a v6 c5 c1 )
        (va_a v7 c1 c2 )
        (va_a v8 c2 c4 )
        (va_a v9 c4 c3 )
        (va_a v10 c3 c5 )
        (esta_en h1 c3 )
        (esta_en h2 c5 )
        (esta_en h3 c1 )
        (esta_en h4 c2 )
        (esta_en h5 c4 )
        (esta_en h6 c3 )
        (esta_en h7 c5 )
        (esta_en h8 c1 )
        (esta_en h9 c2 )
        (esta_en h10 c4 )

    )

    (:goal (and
    (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
    (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
)
```

Output obtenido

```
advancing to distance:    8
                          7
                          5
                          4

best first search space empty! problem proven unsolvable.
```

```
time spent:    0.00 seconds instantiating 60 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 17 facts and 60 actions
               0.00 seconds creating final representation with 16 relevant facts, 2 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 531 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Como podemos observar, el planificador nos indica que no hay solución para el problema, como esperábamos.

5.2.3 Prueba 3

Explicación juego de prueba
Para el último juego de pruebas de esta versión queremos probar que tal funciona el planificador para una instancia grande del problema.

Usaremos 20 ciudades, mínimo 16 ciudades a visitar, 30 vuelos, 30 hoteles, mínimo 3 día por ciudad, máximo 6 días por ciudad y querremos mínimo 50 días de recorrido. **Input (problema)**

```
(define (problem agencia_viaje)
   (:domain agencia_viaje)
   (:objects
      cg1 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18 c19 c20
          - ciudad
      vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20
          v21 v22 v23 v24 v25 v26 v27 v28 v29 v30  - vuelo
      h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21
          h22 h23 h24 h25 h26 h27 h28 h29 h30 - hotel
      dias3 dias4 dias5 dias6 - dias_por_ciudad
   )
   (:init
      (= (num_ciudades_escogidas) 0)
      (= (min_ciudades_a_recoger) 16 )
      (= (min_dias_por_ciudad) 3 )
      (= (max_dias_por_ciudad) 6 )
      (= (num_dias_recorrido) 0)
      (= (min_dias_recorrido) 50 )
      (= (dias_por_ciudad dias3) 3)
      (= (dias_por_ciudad dias4) 4)
      (= (dias_por_ciudad dias5) 5)
      (= (dias_por_ciudad dias6) 6)
      (current_ciudad cg1)
      (ciudad_visitada cg1)
      (va_a vg1 cg1 c1)
      (va_a vg1 cg1 c2)
      (va_a vg1 cg1 c3)
      (va_a vg1 cg1 c4)
      (va_a vg1 cg1 c5)
      (va_a vg1 cg1 c6)
      (va_a vg1 cg1 c7)
      (va_a vg1 cg1 c8)
      (va_a vg1 cg1 c9)
      (va_a vg1 cg1 c10)
      (va_a vg1 cg1 c11)
      (va_a vg1 cg1 c12)
      (va_a vg1 cg1 c13)
      (va_a vg1 cg1 c14)
      (va_a vg1 cg1 c15)
      (va_a vg1 cg1 c16)
      (va_a vg1 cg1 c17)
```

```
(va_a vg1 cg1 c18)
(va_a vg1 cg1 c19)
(va_a vg1 cg1 c20)
(va_a v1 c1 c3 )
(va_a v2 c3 c17 )
(va_a v3 c17 c9 )
(va_a v4 c9 c19 )
(va_a v5 c19 c10 )
(va_a v6 c10 c7 )
(va_a v7 c7 c20 )
(va_a v8 c20 c2 )
(va_a v9 c2 c5 )
(va_a v10 c5 c4 )
(va_a v11 c4 c18 )
(va_a v12 c18 c11 )
(va_a v13 c11 c6 )
(va_a v14 c6 c15 )
(va_a v15 c15 c8 )
(va_a v16 c8 c13 )
(va_a v17 c13 c14 )
(va_a v18 c14 c12 )
(va_a v19 c12 c16 )
(va_a v20 c16 c1 )
(va_a v21 c7 c18 )
(va_a v22 c18 c4 )
(va_a v23 c4 c15 )
(va_a v24 c15 c12 )
(va_a v25 c12 c19 )
(va_a v26 c19 c2 )
(va_a v27 c2 c11 )
(va_a v28 c11 c6 )
(va_a v29 c6 c1 )
(va_a v30 c1 c17 )
(esta_en h1 c13 )
(esta_en h2 c7 )
(esta_en h3 c18 )
(esta_en h4 c4 )
(esta_en h5 c15 )
(esta_en h6 c12 )
(esta_en h7 c19 )
(esta_en h8 c2 )
(esta_en h9 c11 )
(esta_en h10 c6 )
(esta_en h11 c1 )
(esta_en h12 c17 )
(esta_en h13 c9 )
(esta_en h14 c20 )
(esta_en h15 c8 )
(esta_en h16 c5 )
(esta_en h17 c16 )
(esta_en h18 c3 )
(esta_en h19 c14 )
(esta_en h20 c10 )
(esta_en h21 c13 )
(esta_en h22 c7 )
(esta_en h23 c18 )
(esta_en h24 c4 )
(esta_en h25 c15 )
(esta_en h26 c12 )
(esta_en h27 c19 )
(esta_en h28 c2 )
(esta_en h29 c11 )
(esta_en h30 c6 )
```

```
    )

    (:goal (and
    (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
    (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
)
```

Output obtenido

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

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C10 VG1 H20 DIAS3
        1: ANADIR_CIUDAD C10 C7 V6 H22 DIAS3
        2: ANADIR_CIUDAD C7 C20 V7 H14 DIAS3
        3: ANADIR_CIUDAD C20 C2 V8 H28 DIAS3
        4: ANADIR_CIUDAD C2 C5 V9 H16 DIAS3
        5: ANADIR_CIUDAD C5 C4 V10 H24 DIAS3
        6: ANADIR_CIUDAD C4 C18 V11 H3 DIAS3
        7: ANADIR_CIUDAD C18 C11 V12 H29 DIAS3
        8: ANADIR_CIUDAD C11 C6 V13 H30 DIAS3
        9: ANADIR_CIUDAD C6 C15 V14 H5 DIAS3
       10: ANADIR_CIUDAD C15 C8 V15 H15 DIAS3
       11: ANADIR_CIUDAD C8 C13 V16 H21 DIAS3
       12: ANADIR_CIUDAD C13 C14 V17 H19 DIAS3
       13: ANADIR_CIUDAD C14 C12 V18 H26 DIAS3
       14: ANADIR_CIUDAD C12 C16 V19 H17 DIAS3
       15: ANADIR_CIUDAD C16 C1 V20 H11 DIAS5


time spent:    0.00 seconds instantiating 312 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 62 facts and 312 actions
               0.00 seconds creating final representation with 61 relevant facts, 2 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 400 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Como podemos observar en los resultados obtenidos el planificador funciona perfectamente para esta extensión incluso con grandes tamaños de problema. Si nos fijamos se cumplen todas las restricciones necesarias.

Una vez realizados estos tres juegos de prueba podemos asegurar el correcto funcionamiento del planificador para la extensión1.

5.3 Extensión 2: Interés de las ciudades

En la extensión2 hemos añadido un parámetro que indica cuanto interés se tiene en una ciudad, en nuestro script asignamos esta como un número aleatorio entre 0 (mayor interés por visitarla) y el número de ciudades actual (menor interés por visitarla). En estos juegos de pruebas veremos el correcto funcionamiento de esta nueva funcionalidad.

5.3.1 Prueba 1

Explicación juego de prueba
Para empezar usaremos un juego de pruebas pequeño, así comprobaremos si las ciudades con más interés se visitan antes que aquellas con menor interés. Para esto usaremos 8 ciudades, 3 ciudades a visitar como mínimo, 16 vuelos, 10 alojamientos, mínimo de 1 día por ciudad, máximo de 3 días por ciudad y mínimo de 5 días de recorrido. **Input (problema)**

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1  c1  c2  c3  c4  c5  c6  c7  c8   - ciudad
        vg1  v1  v2  v3  v4  v5  v6  v7  v8  v9  v10  v11  v12  v13  v14  v15  v16   - vuelo
        h1  h2  h3  h4  h5  h6  h7  h8  h9  h10 - hotel
        dias1 dias2 dias3 - dias_por_ciudad
    )
    (:init
        (= (num_ciudades_escogidas) 0)
        (= (min_ciudades_a_recoger) 3 )
        (= (min_dias_por_ciudad) 1 )
        (= (max_dias_por_ciudad) 3 )
        (= (num_dias_recorrido) 0)
        (= (min_dias_recorrido) 5 )
        (= (dias_por_ciudad dias1) 1)
        (= (dias_por_ciudad dias2) 2)
        (= (dias_por_ciudad dias3) 3)
         (= (interes_actual) 0)
        (= (interes_ciudad cg1) 3)
        (= (interes_ciudad c1) 7)
        (= (interes_ciudad c2) 4)
        (= (interes_ciudad c3) 3)
        (= (interes_ciudad c4) 4)
        (= (interes_ciudad c5) 1)
        (= (interes_ciudad c6) 0)
        (= (interes_ciudad c7) 1)
        (current_ciudad cg1)
        (= (interes_ciudad cg1) 0)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
        (va_a vg1 cg1 c2)
        (va_a vg1 cg1 c3)
        (va_a vg1 cg1 c4)
        (va_a vg1 cg1 c5)
        (va_a vg1 cg1 c6)
        (va_a vg1 cg1 c7)
        (va_a vg1 cg1 c8)
        (va_a v1 c2 c8 )
        (va_a v2 c8 c5 )
        (va_a v3 c5 c6 )
```

```
        (va_a v4 c6 c3 )
        (va_a v5 c3 c4 )
        (va_a v6 c4 c1 )
        (va_a v7 c1 c7 )
        (va_a v8 c7 c2 )
        (va_a v9 c3 c6 )
        (va_a v10 c6 c4 )
        (va_a v11 c4 c7 )
        (va_a v12 c7 c2 )
        (va_a v13 c2 c5 )
        (va_a v14 c5 c1 )
        (va_a v15 c1 c8 )
        (va_a v16 c8 c3 )
        (esta_en h1 c8 )
        (esta_en h2 c3 )
        (esta_en h3 c6 )
        (esta_en h4 c4 )
        (esta_en h5 c7 )
        (esta_en h6 c2 )
        (esta_en h7 c5 )
        (esta_en h8 c1 )
        (esta_en h9 c8 )
        (esta_en h10 c3 )

    )

    (:goal (and
    (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
    (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
   ;; minimize va DESPUES del goal
   (:metric minimize
      (interes_actual)
   )
)
```

Output obtenido

```
advancing to distance:    3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C3 VG1 H2 DIAS3
        1: ANADIR_CIUDAD C3 C4 V5 H4 DIAS3
        2: ANADIR_CIUDAD C4 C7 V11 H5 DIAS1


time spent:    0.00 seconds instantiating 90 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 26 facts and 72 actions
               0.00 seconds creating final representation with 25 relevant facts, 3 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 36 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Como podemos observar en los resultados obtenidos, visitamos las ciudades C2, C3 y C1, estas son algunas de las ciudades con más interés, la combinación de ciudades escogida depende en parte de la conexión entre ciudades con los vuelos. Aun así podemos ver el correcto funcionamiento de esta extensión. Dado que hemos usado fluentes y hemos pedido que se minimice el interés total de las ciudades, ya que un menor valor de interés total indica un mayor interés de las ciudades, el programa intenta evitar las ciudades con valores de interés muy altos.

5.3.2 Prueba 2

Explicación juego de prueba
Para este segundo juego de pruebas probaremos que el planificador tenga en cuenta ciudades con peor interés si es necesario para llegar al mínimo de días del viaje o al número mínimo de ciudades.

Para esto usaremos 4 ciudades, 3 ciudades a visitar como mínimo, 10 vuelos, 10 alojamientos, mínimo de 1 día por ciudad, máximo de 3 días por ciudad y mínimo de 4 días de recorrido. **Input (problema)**

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1  c1  c2  c3  c4   - ciudad
        vg1  v1  v2  v3  v4  v5  v6  v7  v8  v9  v10   - vuelo
        h1  h2  h3  h4  h5  h6  h7  h8  h9  h10 - hotel
        dias1 dias2 dias3 - dias_por_ciudad
    )
    (:init
        (= (num_ciudades_escogidas) 0)
        (= (min_ciudades_a_recoger) 3 )
        (= (min_dias_por_ciudad) 1 )
        (= (max_dias_por_ciudad) 3 )
        (= (num_dias_recorrido) 0)
        (= (min_dias_recorrido) 4 )
        (= (dias_por_ciudad dias1) 1)
        (= (dias_por_ciudad dias2) 2)
        (= (dias_por_ciudad dias3) 3)
         (= (interes_actual) 0)
        (= (interes_ciudad cg1) 3)
        (= (interes_ciudad c1) 4)
        (= (interes_ciudad c2) 0)
        (= (interes_ciudad c3) 0)
        (current_ciudad cg1)
        (= (interes_ciudad cg1) 0)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
        (va_a vg1 cg1 c2)
        (va_a vg1 cg1 c3)
        (va_a vg1 cg1 c4)
        (va_a v1 c4 c1 )
        (va_a v2 c1 c2 )
        (va_a v3 c2 c3 )
        (va_a v4 c3 c4 )
        (va_a v5 c4 c3 )
        (va_a v6 c3 c2 )
        (va_a v7 c2 c1 )
        (va_a v8 c1 c4 )
        (va_a v9 c4 c3 )
        (va_a v10 c3 c2 )
        (esta_en h1 c1 )
        (esta_en h2 c4 )
        (esta_en h3 c3 )
        (esta_en h4 c2 )
        (esta_en h5 c1 )
        (esta_en h6 c4 )
        (esta_en h7 c3 )
```

```
        (esta_en h8 c2 )
        (esta_en h9 c1 )
        (esta_en h10 c4 )

    )

    (:goal (and
    (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
    (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
   ;; minimize va DESPUES del goal
   (:metric minimize
      (interes_actual)
   )
)
```

Output obtenido

```
advancing to distance:    3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C1 VG1 H1 DIAS1
        1: ANADIR_CIUDAD C1 C2 V2 H4 DIAS1
        2: ANADIR_CIUDAD C2 C3 V3 H3 DIAS2


time spent:    0.00 seconds instantiating 102 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 14 facts and 75 actions
               0.00 seconds creating final representation with 13 relevant facts, 3 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 22 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Como podemos observar en los resultados obtenidos el planificador escoge ciudades con peor interés (c1) para conseguir llegar al mínimo de ciudades y de recorrido deseado.

5.3.3 Prueba 3

Explicación juego de prueba
Para este último juego de pruebas de esta extensión probaremos que tal funciona el planificador con un tamaño del problema grande.

Para esto usaremos 20 ciudades, 16 ciudades a visitar como mínimo, 40 vuelos, 30 alojamientos, mínimo de 3 día por ciudad, máximo de 6 días por ciudad y mínimo de 45 días de recorrido. **Input (problema)**

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18 c19 c20
            - ciudad
        vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20
            v21 v22 v23 v24 v25 v26 v27 v28 v29 v30 v31 v32 v33 v34 v35 v36 v37 v38
            v39 v40  - vuelo
        h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21
            h22 h23 h24 h25 h26 h27 h28 h29 h30 - hotel
```

```
    dias3 dias4 dias5 dias6 - dias_por_ciudad
)
(:init
    (= (num_ciudades_escogidas) 0)
    (= (min_ciudades_a_recoger) 16 )
    (= (min_dias_por_ciudad) 3 )
    (= (max_dias_por_ciudad) 6 )
    (= (num_dias_recorrido) 0)
    (= (min_dias_recorrido) 45 )
    (= (dias_por_ciudad dias3) 3)
    (= (dias_por_ciudad dias4) 4)
    (= (dias_por_ciudad dias5) 5)
    (= (dias_por_ciudad dias6) 6)
     (= (interes_actual) 0)
    (= (interes_ciudad cg1) 14)
    (= (interes_ciudad c1) 19)
    (= (interes_ciudad c2) 2)
    (= (interes_ciudad c3) 11)
    (= (interes_ciudad c4) 18)
    (= (interes_ciudad c5) 11)
    (= (interes_ciudad c6) 19)
    (= (interes_ciudad c7) 2)
    (= (interes_ciudad c8) 7)
    (= (interes_ciudad c9) 9)
    (= (interes_ciudad c10) 8)
    (= (interes_ciudad c11) 0)
    (= (interes_ciudad c12) 6)
    (= (interes_ciudad c13) 8)
    (= (interes_ciudad c14) 19)
    (= (interes_ciudad c15) 7)
    (= (interes_ciudad c16) 6)
    (= (interes_ciudad c17) 18)
    (= (interes_ciudad c18) 19)
    (= (interes_ciudad c19) 20)
    (current_ciudad cg1)
    (= (interes_ciudad cg1) 0)
    (ciudad_visitada cg1)
    (va_a vg1 cg1 c1)
    (va_a vg1 cg1 c2)
    (va_a vg1 cg1 c3)
    (va_a vg1 cg1 c4)
    (va_a vg1 cg1 c5)
    (va_a vg1 cg1 c6)
    (va_a vg1 cg1 c7)
    (va_a vg1 cg1 c8)
    (va_a vg1 cg1 c9)
    (va_a vg1 cg1 c10)
    (va_a vg1 cg1 c11)
    (va_a vg1 cg1 c12)
    (va_a vg1 cg1 c13)
    (va_a vg1 cg1 c14)
    (va_a vg1 cg1 c15)
    (va_a vg1 cg1 c16)
    (va_a vg1 cg1 c17)
    (va_a vg1 cg1 c18)
    (va_a vg1 cg1 c19)
    (va_a vg1 cg1 c20)
    (va_a v1 c8 c18 )
    (va_a v2 c18 c17 )
    (va_a v3 c17 c9 )
    (va_a v4 c9 c1 )
    (va_a v5 c1 c15 )
    (va_a v6 c15 c6 )
```

```
(va_a v7 c6 c12 )
(va_a v8 c12 c7 )
(va_a v9 c7 c14 )
(va_a v10 c14 c2 )
(va_a v11 c2 c5 )
(va_a v12 c5 c19 )
(va_a v13 c19 c13 )
(va_a v14 c13 c20 )
(va_a v15 c20 c4 )
(va_a v16 c4 c10 )
(va_a v17 c10 c11 )
(va_a v18 c11 c3 )
(va_a v19 c3 c16 )
(va_a v20 c16 c8 )
(va_a v21 c9 c20 )
(va_a v22 c20 c18 )
(va_a v23 c18 c3 )
(va_a v24 c3 c17 )
(va_a v25 c17 c16 )
(va_a v26 c16 c5 )
(va_a v27 c5 c1 )
(va_a v28 c1 c6 )
(va_a v29 c6 c19 )
(va_a v30 c19 c15 )
(va_a v31 c15 c14 )
(va_a v32 c14 c11 )
(va_a v33 c11 c13 )
(va_a v34 c13 c12 )
(va_a v35 c12 c10 )
(va_a v36 c10 c4 )
(va_a v37 c4 c8 )
(va_a v38 c8 c7 )
(va_a v39 c7 c2 )
(va_a v40 c2 c9 )
(esta_en h1 c2 )
(esta_en h2 c9 )
(esta_en h3 c20 )
(esta_en h4 c18 )
(esta_en h5 c3 )
(esta_en h6 c17 )
(esta_en h7 c16 )
(esta_en h8 c5 )
(esta_en h9 c1 )
(esta_en h10 c6 )
(esta_en h11 c19 )
(esta_en h12 c15 )
(esta_en h13 c14 )
(esta_en h14 c11 )
(esta_en h15 c13 )
(esta_en h16 c12 )
(esta_en h17 c10 )
(esta_en h18 c4 )
(esta_en h19 c8 )
(esta_en h20 c7 )
(esta_en h21 c2 )
(esta_en h22 c9 )
(esta_en h23 c20 )
(esta_en h24 c18 )
(esta_en h25 c3 )
(esta_en h26 c17 )
(esta_en h27 c16 )
(esta_en h28 c5 )
(esta_en h29 c1 )
```

```
       (esta_en h30 c6 )

    )

    (:goal (and
    (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
    (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
   ;; minimize va DESPUES del goal
   (:metric minimize
      (interes_actual)
   )
)
```

Output obtenido

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

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C19 VG1 H11 DIAS6
        1: ANADIR_CIUDAD C19 C13 V13 H15 DIAS6
        2: ANADIR_CIUDAD C13 C12 V34 H16 DIAS6
        3: ANADIR_CIUDAD C12 C10 V35 H17 DIAS6
        4: ANADIR_CIUDAD C10 C4 V36 H18 DIAS6
        5: ANADIR_CIUDAD C4 C8 V37 H19 DIAS6
        6: ANADIR_CIUDAD C8 C7 V38 H20 DIAS3
        7: ANADIR_CIUDAD C7 C14 V9 H13 DIAS3
        8: ANADIR_CIUDAD C14 C11 V32 H14 DIAS3
        9: ANADIR_CIUDAD C11 C3 V18 H25 DIAS3
       10: ANADIR_CIUDAD C3 C17 V24 H26 DIAS3
       11: ANADIR_CIUDAD C17 C16 V25 H27 DIAS3
       12: ANADIR_CIUDAD C16 C5 V26 H28 DIAS3
       13: ANADIR_CIUDAD C5 C1 V27 H29 DIAS3
       14: ANADIR_CIUDAD C1 C15 V5 H12 DIAS3
       15: ANADIR_CIUDAD C15 C6 V6 H30 DIAS3


time spent:    0.00 seconds instantiating 360 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 62 facts and 336 actions
               0.00 seconds creating final representation with 61 relevant facts, 3 relevant fluents
```

```
0.00 seconds computing LNF
0.01 seconds building connectivity graph
0.01 seconds searching, evaluating 7453 states, to a max depth of 0
0.02 seconds total time
```

Análisis de los resultados Con los resultados obtenidos podemos ver que el planififcador sigue funcionando correctamente incluso con tamaños de entrada grandes para la extensión2.

5.4 Extensión 3: Control del precio del viaje

En la extensión3 de nuestro planificador tendremos en cuenta el precio del viaje, ahora tendremos que devolver un viaje que tenga un precio entre el mínimo y el máximo indicados por el usuario. En estos juegos de prueba buscaremos ver el correcto funcionamiento de estas nuevas funcionalidades.

5.4.1 Prueba 1

Explicación juego de prueba
Para este primer juego de pruebas veremos que el planificador sigue funcionando correctamente para un juego de pruebas pequeño, comprobando además que se respetan las nuevas restricciones añadidas.

Para esto usaremos 4 ciudades, 3 ciudades a visitar como mínimo, 10 vuelos, 10 alojamientos, mínimo de 1 día por ciudad, máximo de 3 días por ciudad, mínimo de 5 días de recorrido, y para acabar un precio mínimo de 100 euros y máximo de 900 euros. **Input (problema)**

```
(define (problem agencia_viaje)
   (:domain agencia_viaje)
   (:objects
      cg1 c1 c2 c3 c4  - ciudad
      vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10  - vuelo
      h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 - hotel
      dias1 dias2 dias3 - dias_por_ciudad
   )
   (:init
      (= (num_ciudades_escogidas) 0)
      (= (min_ciudades_a_recoger) 3 )
      (= (min_dias_por_ciudad) 1 )
      (= (max_dias_por_ciudad) 3 )
      (= (num_dias_recorrido) 0)
      (= (min_dias_recorrido) 5 )
      (= (dias_por_ciudad dias1) 1)
      (= (dias_por_ciudad dias2) 2)
      (= (dias_por_ciudad dias3) 3)
      (= (min_precio_plan) 100)
      (= (max_precio_plan) 900)
      (= (precio_plan) 0)
      (current_ciudad cg1)
      (ciudad_visitada cg1)
      (va_a vg1 cg1 c1)
      (va_a vg1 cg1 c2)
      (va_a vg1 cg1 c3)
      (va_a vg1 cg1 c4)
      (va_a v1 c4 c2 )
      (va_a v2 c2 c1 )
      (va_a v3 c1 c3 )
      (va_a v4 c3 c4 )
      (va_a v5 c4 c1 )
      (va_a v6 c1 c3 )
      (va_a v7 c3 c2 )
      (va_a v8 c2 c4 )
      (va_a v9 c4 c1 )
      (va_a v10 c1 c3 )
      (esta_en h1 c2 )
```

```
        (esta_en h2 c4 )
        (esta_en h3 c1 )
        (esta_en h4 c3 )
        (esta_en h5 c2 )
        (esta_en h6 c4 )
        (esta_en h7 c1 )
        (esta_en h8 c3 )
        (esta_en h9 c2 )
        (esta_en h10 c4 )
        (= (precio_hotel h1) 226)
        (= (precio_hotel h2) 251)
        (= (precio_hotel h3) 133)
        (= (precio_hotel h4) 155)
        (= (precio_hotel h5) 199)
        (= (precio_hotel h6) 74)
        (= (precio_hotel h7) 139)
        (= (precio_hotel h8) 127)
        (= (precio_hotel h9) 116)
        (= (precio_hotel h10) 295)
        (= (precio_vuelo v1) 135)
        (= (precio_vuelo v2) 19)
        (= (precio_vuelo v3) 149)
        (= (precio_vuelo v4) 136)
        (= (precio_vuelo v5) 282)
        (= (precio_vuelo v6) 89)
        (= (precio_vuelo v7) 176)
        (= (precio_vuelo v8) 244)
        (= (precio_vuelo v9) 107)
        (= (precio_vuelo v10) 103)
        (= (precio_vuelo vg1) 0)

    )

    (:goal (and
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (precio_plan)
    )
)
```

Output obtenido

```
advancing to distance:    3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C2 VG1 H9 DIAS1
        1: ANADIR_CIUDAD C2 C1 V2 H3 DIAS1
        2: ANADIR_CIUDAD C1 C3 V6 H8 DIAS3


time spent:    0.00 seconds instantiating 102 easy, 0 hard action templates
```

```
0.00 seconds reachability analysis, yielding 14 facts and 99 actions
0.00 seconds creating final representation with 13 relevant facts, 4 relevant fluents
0.00 seconds computing LNF
0.00 seconds building connectivity graph
0.00 seconds searching, evaluating 62 states, to a max depth of 0
0.00 seconds total time
```

Análisis de los resultados Como podemos observar en los resultados obtenidos si sumamos todos los precios (y contando que si estás 3 días en una ciudad se cobrará el hotel por noche, osea el precio del hotel multiplicado por el número de "noches"), los resultados obtenidos respetan todas las restricciones.

5.4.2 Prueba 2

Explicación juego de prueba
Para este juego de pruebas probaremos el caso en el que no se pueda realizar el viaje por falta de presupuesto.

Para esto usaremos 6 ciudades, 5 ciudades a visitar como mínimo, 20 vuelos, 20 alojamientos, mínimo de 2 día por ciudad, máximo de 5 días por ciudad, mínimo de 8 días de recorrido, y para acabar un precio mínimo de 10 euros y máximo de 100 euros. **Input (problema)**

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1  c1  c2  c3  c4  c5  c6   - ciudad
        vg1  v1  v2  v3  v4  v5  v6  v7  v8  v9  v10  v11  v12  v13  v14  v15  v16  v17  v18  v19  v20
            - vuelo
        h1  h2  h3  h4  h5  h6  h7  h8  h9  h10  h11  h12  h13  h14  h15  h16  h17  h18  h19  h20  -
            hotel
        dias2 dias3 dias4 dias5 - dias_por_ciudad
    )
    (:init
        (= (num_ciudades_escogidas) 0)
        (= (min_ciudades_a_recoger) 5 )
        (= (min_dias_por_ciudad) 2 )
        (= (max_dias_por_ciudad) 5 )
        (= (num_dias_recorrido) 0)
        (= (min_dias_recorrido) 8 )
        (= (dias_por_ciudad dias2) 2)
        (= (dias_por_ciudad dias3) 3)
        (= (dias_por_ciudad dias4) 4)
        (= (dias_por_ciudad dias5) 5)
        (= (min_precio_plan) 10)
        (= (max_precio_plan) 100)
        (= (precio_plan) 0)
        (current_ciudad cg1)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
        (va_a vg1 cg1 c2)
        (va_a vg1 cg1 c3)
        (va_a vg1 cg1 c4)
        (va_a vg1 cg1 c5)
        (va_a vg1 cg1 c6)
        (va_a v1 c6 c4 )
        (va_a v2 c4 c5 )
        (va_a v3 c5 c2 )
        (va_a v4 c2 c1 )
        (va_a v5 c1 c3 )
        (va_a v6 c3 c6 )
        (va_a v7 c5 c4 )
        (va_a v8 c4 c1 )
        (va_a v9 c1 c6 )
```

```
(va_a v10 c6 c2 )
(va_a v11 c2 c3 )
(va_a v12 c3 c5 )
(va_a v13 c5 c4 )
(va_a v14 c4 c1 )
(va_a v15 c1 c6 )
(va_a v16 c6 c2 )
(va_a v17 c2 c3 )
(va_a v18 c3 c5 )
(va_a v19 c5 c4 )
(va_a v20 c4 c1 )
(esta_en h1 c3 )
(esta_en h2 c5 )
(esta_en h3 c4 )
(esta_en h4 c1 )
(esta_en h5 c6 )
(esta_en h6 c2 )
(esta_en h7 c3 )
(esta_en h8 c5 )
(esta_en h9 c4 )
(esta_en h10 c1 )
(esta_en h11 c6 )
(esta_en h12 c2 )
(esta_en h13 c3 )
(esta_en h14 c5 )
(esta_en h15 c4 )
(esta_en h16 c1 )
(esta_en h17 c6 )
(esta_en h18 c2 )
(esta_en h19 c3 )
(esta_en h20 c5 )
(= (precio_hotel h1) 128)
(= (precio_hotel h2) 38)
(= (precio_hotel h3) 240)
(= (precio_hotel h4) 19)
(= (precio_hotel h5) 175)
(= (precio_hotel h6) 300)
(= (precio_hotel h7) 277)
(= (precio_hotel h8) 280)
(= (precio_hotel h9) 285)
(= (precio_hotel h10) 243)
(= (precio_hotel h11) 255)
(= (precio_hotel h12) 153)
(= (precio_hotel h13) 183)
(= (precio_hotel h14) 238)
(= (precio_hotel h15) 33)
(= (precio_hotel h16) 138)
(= (precio_hotel h17) 76)
(= (precio_hotel h18) 208)
(= (precio_hotel h19) 58)
(= (precio_hotel h20) 152)
(= (precio_vuelo v1) 22)
(= (precio_vuelo v2) 105)
(= (precio_vuelo v3) 125)
(= (precio_vuelo v4) 276)
(= (precio_vuelo v5) 133)
(= (precio_vuelo v6) 61)
(= (precio_vuelo v7) 165)
(= (precio_vuelo v8) 223)
(= (precio_vuelo v9) 192)
(= (precio_vuelo v10) 257)
(= (precio_vuelo v11) 60)
(= (precio_vuelo v12) 150)
```

```
        (= (precio_vuelo v13) 237)
        (= (precio_vuelo v14) 166)
        (= (precio_vuelo v15) 43)
        (= (precio_vuelo v16) 205)
        (= (precio_vuelo v17) 238)
        (= (precio_vuelo v18) 128)
        (= (precio_vuelo v19) 265)
        (= (precio_vuelo v20) 68)
        (= (precio_vuelo vg1) 0)

    )

    (:goal (and
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (precio_plan)
    )
)
```

Output obtenido

```
advancing to distance:    5

best first search space empty! problem proven unsolvable.



time spent:    0.00 seconds instantiating 344 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 20 facts and 8 actions
               0.00 seconds creating final representation with 19 relevant facts, 4 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 8 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Como podemos observar en los resultados obtenidos el viaje no se puede realizar, tal y como esperábamos.

5.4.3 Prueba 3

Explicación juego de prueba
Para el último juego de pruebas de esta extensión usaremos un juego de pruebas grande, para comprobar el correcto funcionamiento del planificador con una entrada grande.

Para esto usaremos 20 ciudades, 16 ciudades a visitar como mínimo, 40 vuelos, 30 alojamientos, mínimo de 2 día por ciudad, máximo de 6 días por ciudad, mínimo de 45 días de recorrido, y para acabar un precio mínimo de 100 euros y máximo de 100000 euros. **Input (problema)**

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18 c19 c20
            - ciudad
```

```
    vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20
        v21 v22 v23 v24 v25 v26 v27 v28 v29 v30 v31 v32 v33 v34 v35 v36 v37 v38
        v39 v40  - vuelo
    h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21
        h22 h23 h24 h25 h26 h27 h28 h29 h30 - hotel
    dias2 dias3 dias4 dias5 dias6 - dias_por_ciudad
)
(:init
    (= (num_ciudades_escogidas) 0)
    (= (min_ciudades_a_recoger) 16 )
    (= (min_dias_por_ciudad) 2 )
    (= (max_dias_por_ciudad) 6 )
    (= (num_dias_recorrido) 0)
    (= (min_dias_recorrido) 45 )
    (= (dias_por_ciudad dias2) 2)
    (= (dias_por_ciudad dias3) 3)
    (= (dias_por_ciudad dias4) 4)
    (= (dias_por_ciudad dias5) 5)
    (= (dias_por_ciudad dias6) 6)
    (= (min_precio_plan) 100)
    (= (max_precio_plan) 10000)
    (= (precio_plan) 0)
    (current_ciudad cg1)
    (ciudad_visitada cg1)
    (va_a vg1 cg1 c1)
    (va_a vg1 cg1 c2)
    (va_a vg1 cg1 c3)
    (va_a vg1 cg1 c4)
    (va_a vg1 cg1 c5)
    (va_a vg1 cg1 c6)
    (va_a vg1 cg1 c7)
    (va_a vg1 cg1 c8)
    (va_a vg1 cg1 c9)
    (va_a vg1 cg1 c10)
    (va_a vg1 cg1 c11)
    (va_a vg1 cg1 c12)
    (va_a vg1 cg1 c13)
    (va_a vg1 cg1 c14)
    (va_a vg1 cg1 c15)
    (va_a vg1 cg1 c16)
    (va_a vg1 cg1 c17)
    (va_a vg1 cg1 c18)
    (va_a vg1 cg1 c19)
    (va_a vg1 cg1 c20)
    (va_a v1 c2 c17 )
    (va_a v2 c17 c10 )
    (va_a v3 c10 c19 )
    (va_a v4 c19 c13 )
    (va_a v5 c13 c12 )
    (va_a v6 c12 c5 )
    (va_a v7 c5 c1 )
    (va_a v8 c1 c6 )
    (va_a v9 c6 c16 )
    (va_a v10 c16 c14 )
    (va_a v11 c14 c15 )
    (va_a v12 c15 c18 )
    (va_a v13 c18 c9 )
    (va_a v14 c9 c7 )
    (va_a v15 c7 c11 )
    (va_a v16 c11 c4 )
    (va_a v17 c4 c20 )
    (va_a v18 c20 c3 )
    (va_a v19 c3 c8 )
```

```
(va_a v20 c8 c2 )
(va_a v21 c3 c14 )
(va_a v22 c14 c9 )
(va_a v23 c9 c12 )
(va_a v24 c12 c11 )
(va_a v25 c11 c4 )
(va_a v26 c4 c13 )
(va_a v27 c13 c18 )
(va_a v28 c18 c10 )
(va_a v29 c10 c20 )
(va_a v30 c20 c1 )
(va_a v31 c1 c17 )
(va_a v32 c17 c7 )
(va_a v33 c7 c6 )
(va_a v34 c6 c2 )
(va_a v35 c2 c15 )
(va_a v36 c15 c8 )
(va_a v37 c8 c19 )
(va_a v38 c19 c16 )
(va_a v39 c16 c5 )
(va_a v40 c5 c3 )
(esta_en h1 c5 )
(esta_en h2 c3 )
(esta_en h3 c14 )
(esta_en h4 c9 )
(esta_en h5 c12 )
(esta_en h6 c11 )
(esta_en h7 c4 )
(esta_en h8 c13 )
(esta_en h9 c18 )
(esta_en h10 c10 )
(esta_en h11 c20 )
(esta_en h12 c1 )
(esta_en h13 c17 )
(esta_en h14 c7 )
(esta_en h15 c6 )
(esta_en h16 c2 )
(esta_en h17 c15 )
(esta_en h18 c8 )
(esta_en h19 c19 )
(esta_en h20 c16 )
(esta_en h21 c5 )
(esta_en h22 c3 )
(esta_en h23 c14 )
(esta_en h24 c9 )
(esta_en h25 c12 )
(esta_en h26 c11 )
(esta_en h27 c4 )
(esta_en h28 c13 )
(esta_en h29 c18 )
(esta_en h30 c10 )
(= (precio_hotel h1) 128)
(= (precio_hotel h2) 229)
(= (precio_hotel h3) 88)
(= (precio_hotel h4) 143)
(= (precio_hotel h5) 28)
(= (precio_hotel h6) 113)
(= (precio_hotel h7) 144)
(= (precio_hotel h8) 279)
(= (precio_hotel h9) 241)
(= (precio_hotel h10) 212)
(= (precio_hotel h11) 31)
(= (precio_hotel h12) 222)
```

```
    (= (precio_hotel h13) 270)
    (= (precio_hotel h14) 214)
    (= (precio_hotel h15) 245)
    (= (precio_hotel h16) 85)
    (= (precio_hotel h17) 259)
    (= (precio_hotel h18) 102)
    (= (precio_hotel h19) 114)
    (= (precio_hotel h20) 229)
    (= (precio_hotel h21) 112)
    (= (precio_hotel h22) 60)
    (= (precio_hotel h23) 236)
    (= (precio_hotel h24) 275)
    (= (precio_hotel h25) 95)
    (= (precio_hotel h26) 278)
    (= (precio_hotel h27) 93)
    (= (precio_hotel h28) 10)
    (= (precio_hotel h29) 111)
    (= (precio_hotel h30) 24)
    (= (precio_vuelo v1) 73)
    (= (precio_vuelo v2) 152)
    (= (precio_vuelo v3) 275)
    (= (precio_vuelo v4) 155)
    (= (precio_vuelo v5) 24)
    (= (precio_vuelo v6) 170)
    (= (precio_vuelo v7) 136)
    (= (precio_vuelo v8) 277)
    (= (precio_vuelo v9) 244)
    (= (precio_vuelo v10) 159)
    (= (precio_vuelo v11) 205)
    (= (precio_vuelo v12) 209)
    (= (precio_vuelo v13) 180)
    (= (precio_vuelo v14) 103)
    (= (precio_vuelo v15) 77)
    (= (precio_vuelo v16) 206)
    (= (precio_vuelo v17) 264)
    (= (precio_vuelo v18) 270)
    (= (precio_vuelo v19) 130)
    (= (precio_vuelo v20) 153)
    (= (precio_vuelo v21) 49)
    (= (precio_vuelo v22) 44)
    (= (precio_vuelo v23) 202)
    (= (precio_vuelo v24) 167)
    (= (precio_vuelo v25) 178)
    (= (precio_vuelo v26) 26)
    (= (precio_vuelo v27) 101)
    (= (precio_vuelo v28) 84)
    (= (precio_vuelo v29) 69)
    (= (precio_vuelo v30) 113)
    (= (precio_vuelo v31) 271)
    (= (precio_vuelo v32) 127)
    (= (precio_vuelo v33) 181)
    (= (precio_vuelo v34) 251)
    (= (precio_vuelo v35) 182)
    (= (precio_vuelo v36) 293)
    (= (precio_vuelo v37) 16)
    (= (precio_vuelo v38) 144)
    (= (precio_vuelo v39) 262)
    (= (precio_vuelo v40) 89)
    (= (precio_vuelo vg1) 0)

)

(:goal (and
```

```
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (precio_plan)
    )
)
```

Output obtenido

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

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C4 VG1 H27 DIAS2
        1: ANADIR_CIUDAD C4 C13 V26 H28 DIAS6
        2: ANADIR_CIUDAD C13 C18 V27 H29 DIAS2
        3: ANADIR_CIUDAD C18 C10 V28 H30 DIAS6
        4: ANADIR_CIUDAD C10 C20 V29 H11 DIAS5
        5: ANADIR_CIUDAD C20 C3 V18 H22 DIAS6
        6: ANADIR_CIUDAD C3 C14 V21 H3 DIAS6
        7: ANADIR_CIUDAD C14 C15 V11 H17 DIAS2
        8: ANADIR_CIUDAD C15 C8 V36 H18 DIAS2
        9: ANADIR_CIUDAD C8 C19 V37 H19 DIAS2
       10: ANADIR_CIUDAD C19 C16 V38 H20 DIAS2
       11: ANADIR_CIUDAD C16 C5 V39 H21 DIAS2
       12: ANADIR_CIUDAD C5 C1 V7 H12 DIAS2
       13: ANADIR_CIUDAD C1 C6 V8 H15 DIAS2
       14: ANADIR_CIUDAD C6 C2 V34 H16 DIAS2
       15: ANADIR_CIUDAD C2 C17 V1 H13 DIAS2


time spent:    0.00 seconds instantiating 450 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 62 facts and 450 actions
               0.00 seconds creating final representation with 61 relevant facts, 4 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
             132.04 seconds searching, evaluating 118666 states, to a max depth of 0
```

```
        132.04 seconds total time
```

Análisis de los resultados Como podemos observar en los resultados obtenidos, el planificador cumple todas las restricciones pero tarda un poco más en calcular una solución, esto es normal debido al gran tamaño del problema.

5.5 Extensión 4: Ponderación de interés y precios

En la extensión 4 combinamos las extensiones 2 y 3 (ya probadas anteriormente) ponderando el interés y los precios, en esta sección probaremos el correcto funcionamiento de esta extensión de manera similar a las anteriores.

5.5.1 Prueba 1

Explicación juego de prueba
En el primer juego de prueba comprobaremos que para una entrada pequeña el planificador cumple todas las restricciones (incluida la ponderación de precio e interés).

Para esto usaremos 4 ciudades, 3 ciudades a visitar como mínimo, 10 vuelos, 10 alojamientos, mínimo de 1 día por ciudad, máximo de 3 días por ciudad, mínimo de 5 días de recorrido, y para acabar un precio mínimo de 100 euros y máximo de 500 euros. **Input (problema)**

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1  c1  c2  c3  c4   - ciudad
        vg1  v1  v2  v3  v4  v5  v6  v7  v8  v9  v10   - vuelo
        h1  h2  h3  h4  h5  h6  h7  h8  h9  h10  - hotel
        dias1 dias2 dias3 - dias_por_ciudad
    )
    (:init
        (= (num_ciudades_escogidas) 0)
        (= (min_ciudades_a_recoger) 3 )
        (= (min_dias_por_ciudad) 1 )
        (= (max_dias_por_ciudad) 3 )
        (= (num_dias_recorrido) 0)
        (= (min_dias_recorrido) 5 )
        (= (dias_por_ciudad dias1) 1)
        (= (dias_por_ciudad dias2) 2)
        (= (dias_por_ciudad dias3) 3)
        (= (interes_ciudad cg1) 0)
        (= (interes_ciudad c1) 4)
        (= (interes_ciudad c2) 1)
        (= (interes_ciudad c3) 0)
        (= (min_precio_plan) 100)
        (= (max_precio_plan) 500)
        (= (precio_plan) 0)
        (current_ciudad cg1)
        (= (interes_ciudad cg1) 0)
        (= (interes_actual) 0)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
        (va_a vg1 cg1 c2)
        (va_a vg1 cg1 c3)
        (va_a vg1 cg1 c4)
        (va_a v1 c2 c3 )
        (va_a v2 c3 c1 )
        (va_a v3 c1 c4 )
        (va_a v4 c4 c2 )
        (va_a v5 c2 c4 )
        (va_a v6 c4 c1 )
        (va_a v7 c1 c3 )
```

```
        (va_a v8 c3 c2 )
        (va_a v9 c2 c4 )
        (va_a v10 c4 c1 )
        (esta_en h1 c3 )
        (esta_en h2 c2 )
        (esta_en h3 c4 )
        (esta_en h4 c1 )
        (esta_en h5 c3 )
        (esta_en h6 c2 )
        (esta_en h7 c4 )
        (esta_en h8 c1 )
        (esta_en h9 c3 )
        (esta_en h10 c2 )
        (= (precio_hotel h1) 137)
        (= (precio_hotel h2) 153)
        (= (precio_hotel h3) 112)
        (= (precio_hotel h4) 109)
        (= (precio_hotel h5) 64)
        (= (precio_hotel h6) 272)
        (= (precio_hotel h7) 41)
        (= (precio_hotel h8) 44)
        (= (precio_hotel h9) 43)
        (= (precio_hotel h10) 35)
        (= (precio_vuelo v1) 217)
        (= (precio_vuelo v2) 246)
        (= (precio_vuelo v3) 29)
        (= (precio_vuelo v4) 26)
        (= (precio_vuelo v5) 59)
        (= (precio_vuelo v6) 208)
        (= (precio_vuelo v7) 60)
        (= (precio_vuelo v8) 111)
        (= (precio_vuelo v9) 37)
        (= (precio_vuelo v10) 107)
        (= (precio_vuelo vg1) 0)

    )

    (:goal (and
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (+ (precio_plan) (* (interes_actual) 400))
    )
)
```

Output obtenido

```
advancing to distance:    3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C1 VG1 H8 DIAS3
        1: ANADIR_CIUDAD C1 C3 V7 H9 DIAS1
```

```
        2: ANADIR_CIUDAD C3 C2 V8 H10 DIAS1


time spent:    0.00 seconds instantiating 102 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 14 facts and 68 actions
               0.00 seconds creating final representation with 13 relevant facts, 5 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 72 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Con los resultados obtenidos podemos observar como el planificador funciona correctamente, aún así con este juego de pruebas no podemos ver si se cogen las ciudades con más interés, en lo siguientes juegos lo comprobaremos.

5.5.2 Prueba 2

Explicación juego de prueba
En este segundo juego de pruebas vamos a ver que el planificador escoge las ciudades con mayor interés.

Para esto usaremos 10 ciudades, 5 ciudades a visitar como mínimo, 20 vuelos, 20 alojamientos, mínimo de 1 día por ciudad, máximo de 3 días por ciudad, mínimo de 5 días de recorrido, y para acabar un precio mínimo de 100 euros y máximo de 800 euros. **Input (problema)**

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10  - ciudad
        vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20
            - vuelo
        h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 -
            hotel
        dias1 dias2 dias3 - dias_por_ciudad
    )
    (:init
        (= (num_ciudades_escogidas) 0)
        (= (min_ciudades_a_recoger) 5 )
        (= (min_dias_por_ciudad) 1 )
        (= (max_dias_por_ciudad) 3 )
        (= (num_dias_recorrido) 0)
        (= (min_dias_recorrido) 5 )
        (= (dias_por_ciudad dias1) 1)
        (= (dias_por_ciudad dias2) 2)
        (= (dias_por_ciudad dias3) 3)
        (= (interes_ciudad cg1) 2)
        (= (interes_ciudad c1) 6)
        (= (interes_ciudad c2) 0)
        (= (interes_ciudad c3) 0)
        (= (interes_ciudad c4) 0)
        (= (interes_ciudad c5) 5)
        (= (interes_ciudad c6) 0)
        (= (interes_ciudad c7) 8)
        (= (interes_ciudad c8) 8)
        (= (interes_ciudad c9) 2)
        (= (min_precio_plan) 100)
        (= (max_precio_plan) 800)
        (= (precio_plan) 0)
        (current_ciudad cg1)
        (= (interes_ciudad cg1) 0)
        (= (interes_actual) 0)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
```

```
(va_a vg1 cg1 c2)
(va_a vg1 cg1 c3)
(va_a vg1 cg1 c4)
(va_a vg1 cg1 c5)
(va_a vg1 cg1 c6)
(va_a vg1 cg1 c7)
(va_a vg1 cg1 c8)
(va_a vg1 cg1 c9)
(va_a vg1 cg1 c10)
(va_a v1 c7 c10 )
(va_a v2 c10 c5 )
(va_a v3 c5 c3 )
(va_a v4 c3 c1 )
(va_a v5 c1 c4 )
(va_a v6 c4 c9 )
(va_a v7 c9 c8 )
(va_a v8 c8 c2 )
(va_a v9 c2 c6 )
(va_a v10 c6 c7 )
(va_a v11 c7 c4 )
(va_a v12 c4 c9 )
(va_a v13 c9 c8 )
(va_a v14 c8 c5 )
(va_a v15 c5 c3 )
(va_a v16 c3 c2 )
(va_a v17 c2 c6 )
(va_a v18 c6 c10 )
(va_a v19 c10 c1 )
(va_a v20 c1 c7 )
(esta_en h1 c1 )
(esta_en h2 c7 )
(esta_en h3 c4 )
(esta_en h4 c9 )
(esta_en h5 c8 )
(esta_en h6 c5 )
(esta_en h7 c3 )
(esta_en h8 c2 )
(esta_en h9 c6 )
(esta_en h10 c10 )
(esta_en h11 c1 )
(esta_en h12 c7 )
(esta_en h13 c4 )
(esta_en h14 c9 )
(esta_en h15 c8 )
(esta_en h16 c5 )
(esta_en h17 c3 )
(esta_en h18 c2 )
(esta_en h19 c6 )
(esta_en h20 c10 )
(= (precio_hotel h1) 54)
(= (precio_hotel h2) 44)
(= (precio_hotel h3) 120)
(= (precio_hotel h4) 153)
(= (precio_hotel h5) 206)
(= (precio_hotel h6) 195)
(= (precio_hotel h7) 143)
(= (precio_hotel h8) 163)
(= (precio_hotel h9) 276)
(= (precio_hotel h10) 36)
(= (precio_hotel h11) 194)
(= (precio_hotel h12) 101)
(= (precio_hotel h13) 241)
(= (precio_hotel h14) 189)
```

```
        (= (precio_hotel h15) 216)
        (= (precio_hotel h16) 156)
        (= (precio_hotel h17) 37)
        (= (precio_hotel h18) 295)
        (= (precio_hotel h19) 45)
        (= (precio_hotel h20) 78)
        (= (precio_vuelo v1) 73)
        (= (precio_vuelo v2) 189)
        (= (precio_vuelo v3) 235)
        (= (precio_vuelo v4) 177)
        (= (precio_vuelo v5) 126)
        (= (precio_vuelo v6) 280)
        (= (precio_vuelo v7) 151)
        (= (precio_vuelo v8) 175)
        (= (precio_vuelo v9) 18)
        (= (precio_vuelo v10) 20)
        (= (precio_vuelo v11) 77)
        (= (precio_vuelo v12) 50)
        (= (precio_vuelo v13) 12)
        (= (precio_vuelo v14) 225)
        (= (precio_vuelo v15) 179)
        (= (precio_vuelo v16) 35)
        (= (precio_vuelo v17) 138)
        (= (precio_vuelo v18) 51)
        (= (precio_vuelo v19) 193)
        (= (precio_vuelo v20) 107)
        (= (precio_vuelo vg1) 0)

    )

    (:goal (and
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (+ (precio_plan) (* (interes_actual) 400))
    )
)
```

Output obtenido

```
advancing to distance:    5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C5 VG1 H16 DIAS1
        1: ANADIR_CIUDAD C5 C3 V15 H17 DIAS1
        2: ANADIR_CIUDAD C3 C2 V16 H8 DIAS1
        3: ANADIR_CIUDAD C2 C6 V9 H19 DIAS1
        4: ANADIR_CIUDAD C6 C7 V10 H2 DIAS1
```

```
time spent:    0.00 seconds instantiating 180 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 32 facts and 153 actions
               0.00 seconds creating final representation with 31 relevant facts, 5 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 258 states, to a max depth of 0
               0.00 seconds total time
```

Análisis de los resultados Observando los resultados obtenidos podemos ver como se priorizan las ciudades con mejor interés.

5.5.3 Prueba 3

Explicación juego de prueba
Para este último juego de pruebas vamos a dar una entrada MUY grande al problema (no hemos podido poner una entrada de mayor tamaño debido a la restricción de tamaño del ff) para así ver como se comporta el planificador.

Para esto usaremos 20 ciudades, 15 ciudades a visitar como mínimo, 25 vuelos, 25 alojamientos, mínimo de 1 día por ciudad, máximo de 4 días por ciudad, mínimo de 30 días de recorrido, y para acabar un precio mínimo de 500 euros y máximo de 100000 euros. **Input (problema)**

```
(define (problem agencia_viaje)
   (:domain agencia_viaje)
   (:objects
      cg1 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18 c19 c20
          - ciudad
      vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20
          v21 v22 v23 v24 v25  - vuelo
      h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21
          h22 h23 h24 h25 - hotel
      dias1 dias2 dias3 dias4 - dias_por_ciudad
   )
   (:init
      (= (num_ciudades_escogidas) 0)
      (= (min_ciudades_a_recoger) 15 )
      (= (min_dias_por_ciudad) 1 )
      (= (max_dias_por_ciudad) 4 )
      (= (num_dias_recorrido) 0)
      (= (min_dias_recorrido) 30 )
      (= (dias_por_ciudad dias1) 1)
      (= (dias_por_ciudad dias2) 2)
      (= (dias_por_ciudad dias3) 3)
      (= (dias_por_ciudad dias4) 4)
      (= (interes_ciudad cg1) 7)
      (= (interes_ciudad c1) 10)
      (= (interes_ciudad c2) 7)
      (= (interes_ciudad c3) 5)
      (= (interes_ciudad c4) 9)
      (= (interes_ciudad c5) 1)
      (= (interes_ciudad c6) 12)
      (= (interes_ciudad c7) 7)
      (= (interes_ciudad c8) 6)
      (= (interes_ciudad c9) 17)
      (= (interes_ciudad c10) 15)
      (= (interes_ciudad c11) 18)
      (= (interes_ciudad c12) 10)
      (= (interes_ciudad c13) 4)
      (= (interes_ciudad c14) 8)
      (= (interes_ciudad c15) 20)
      (= (interes_ciudad c16) 9)
```

```
(= (interes_ciudad c17) 13)
(= (interes_ciudad c18) 17)
(= (interes_ciudad c19) 18)
(= (min_precio_plan) 400)
(= (max_precio_plan) 100000)
(= (precio_plan) 0)
(current_ciudad cg1)
(= (interes_ciudad cg1) 0)
(= (interes_actual) 0)
(ciudad_visitada cg1)
(va_a vg1 cg1 c1)
(va_a vg1 cg1 c2)
(va_a vg1 cg1 c3)
(va_a vg1 cg1 c4)
(va_a vg1 cg1 c5)
(va_a vg1 cg1 c6)
(va_a vg1 cg1 c7)
(va_a vg1 cg1 c8)
(va_a vg1 cg1 c9)
(va_a vg1 cg1 c10)
(va_a vg1 cg1 c11)
(va_a vg1 cg1 c12)
(va_a vg1 cg1 c13)
(va_a vg1 cg1 c14)
(va_a vg1 cg1 c15)
(va_a vg1 cg1 c16)
(va_a vg1 cg1 c17)
(va_a vg1 cg1 c18)
(va_a vg1 cg1 c19)
(va_a vg1 cg1 c20)
(va_a v1 c12 c11 )
(va_a v2 c11 c6 )
(va_a v3 c6 c17 )
(va_a v4 c17 c1 )
(va_a v5 c1 c13 )
(va_a v6 c13 c16 )
(va_a v7 c16 c2 )
(va_a v8 c2 c9 )
(va_a v9 c9 c14 )
(va_a v10 c14 c19 )
(va_a v11 c19 c4 )
(va_a v12 c4 c10 )
(va_a v13 c10 c15 )
(va_a v14 c15 c7 )
(va_a v15 c7 c18 )
(va_a v16 c18 c20 )
(va_a v17 c20 c5 )
(va_a v18 c5 c3 )
(va_a v19 c3 c8 )
(va_a v20 c8 c12 )
(va_a v21 c12 c2 )
(va_a v22 c2 c5 )
(va_a v23 c5 c7 )
(va_a v24 c7 c16 )
(va_a v25 c16 c1 )
(esta_en h1 c20 )
(esta_en h2 c12 )
(esta_en h3 c2 )
(esta_en h4 c5 )
(esta_en h5 c7 )
(esta_en h6 c16 )
(esta_en h7 c1 )
(esta_en h8 c13 )
```

```
(esta_en h9 c8 )
(esta_en h10 c17 )
(esta_en h11 c9 )
(esta_en h12 c6 )
(esta_en h13 c10 )
(esta_en h14 c18 )
(esta_en h15 c11 )
(esta_en h16 c3 )
(esta_en h17 c19 )
(esta_en h18 c15 )
(esta_en h19 c14 )
(esta_en h20 c4 )
(esta_en h21 c20 )
(esta_en h22 c12 )
(esta_en h23 c2 )
(esta_en h24 c5 )
(esta_en h25 c7 )
(= (precio_hotel h1) 11)
(= (precio_hotel h2) 129)
(= (precio_hotel h3) 184)
(= (precio_hotel h4) 25)
(= (precio_hotel h5) 290)
(= (precio_hotel h6) 155)
(= (precio_hotel h7) 117)
(= (precio_hotel h8) 293)
(= (precio_hotel h9) 96)
(= (precio_hotel h10) 76)
(= (precio_hotel h11) 131)
(= (precio_hotel h12) 88)
(= (precio_hotel h13) 235)
(= (precio_hotel h14) 273)
(= (precio_hotel h15) 261)
(= (precio_hotel h16) 62)
(= (precio_hotel h17) 62)
(= (precio_hotel h18) 288)
(= (precio_hotel h19) 246)
(= (precio_hotel h20) 29)
(= (precio_hotel h21) 149)
(= (precio_hotel h22) 128)
(= (precio_hotel h23) 35)
(= (precio_hotel h24) 25)
(= (precio_hotel h25) 50)
(= (precio_vuelo v1) 183)
(= (precio_vuelo v2) 293)
(= (precio_vuelo v3) 145)
(= (precio_vuelo v4) 148)
(= (precio_vuelo v5) 190)
(= (precio_vuelo v6) 259)
(= (precio_vuelo v7) 223)
(= (precio_vuelo v8) 174)
(= (precio_vuelo v9) 170)
(= (precio_vuelo v10) 289)
(= (precio_vuelo v11) 44)
(= (precio_vuelo v12) 134)
(= (precio_vuelo v13) 142)
(= (precio_vuelo v14) 131)
(= (precio_vuelo v15) 250)
(= (precio_vuelo v16) 275)
(= (precio_vuelo v17) 92)
(= (precio_vuelo v18) 207)
(= (precio_vuelo v19) 122)
(= (precio_vuelo v20) 261)
(= (precio_vuelo v21) 84)
```

```
        (= (precio_vuelo v22) 135)
        (= (precio_vuelo v23) 60)
        (= (precio_vuelo v24) 189)
        (= (precio_vuelo v25) 224)
        (= (precio_vuelo vg1) 0)

    )

    (:goal (and
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (+ (precio_plan) (* (interes_actual) 400))
    )
)
```

Output obtenido

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

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C5 VG1 H4 DIAS1
        1: ANADIR_CIUDAD C5 C3 V18 H16 DIAS1
        2: ANADIR_CIUDAD C3 C8 V19 H9 DIAS1
        3: ANADIR_CIUDAD C8 C12 V20 H22 DIAS1
        4: ANADIR_CIUDAD C12 C11 V1 H15 DIAS1
        5: ANADIR_CIUDAD C11 C6 V2 H12 DIAS4
        6: ANADIR_CIUDAD C6 C17 V3 H10 DIAS4
        7: ANADIR_CIUDAD C17 C1 V4 H7 DIAS1
        8: ANADIR_CIUDAD C1 C13 V5 H8 DIAS1
        9: ANADIR_CIUDAD C13 C16 V6 H6 DIAS1
       10: ANADIR_CIUDAD C16 C2 V7 H23 DIAS4
       11: ANADIR_CIUDAD C2 C9 V8 H11 DIAS2
       12: ANADIR_CIUDAD C9 C14 V9 H19 DIAS1
       13: ANADIR_CIUDAD C14 C19 V10 H17 DIAS3
       14: ANADIR_CIUDAD C19 C4 V11 H20 DIAS4
```

```
time spent:    0.00 seconds instantiating 232 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 62 facts and 216 actions
               0.00 seconds creating final representation with 61 relevant facts, 5 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
             299.87 seconds searching, evaluating 1516746 states, to a max depth of 0
             299.87 seconds total time
```

Análisis de los resultados Como podemos ver el planificador tarda 300 segundos para esta instancia del problema, tal y como esperábamos con entradas grandes es más lento, pero aún así funciona correctamente.

6 Extra 1

En el extra 1 se nos pedía crear un programa que generara juegos de prueba generados aleatoriamente y que los usáramos en las pruebas. Hemos creado 5 programas, una por cada expansión del proyecto que nos generan, dados unos parámetros como el mínimo de ciudades a visitar, el presupuesto y demás un juego de pruebas en formato PDDL listo para que lo lea el Metric-FF.

7 Extra 2

El extra 2 consiste en usar el generador para la extensión 3 con diferentes parámetros para generar problemas con un tamaño creciente. Usaremos el número de instancias y el número de líneas de inicialización como indicadores del tamaño del problema.

Aquí mostramos los parámetros usados como input:

```
//////////////////
EXTRA1
//////////////////
Inserte el numero de ciudades a generar
3
Inserte el número mínimo de ciudades a visitar
3
Inserte el numero de vuelos a generar
Tiene que ser mayor o igual al numero de ciudades
5
Inserte el numero de alojamientos a generar
Tiene que ser mayor o igual al numero de ciudades
5
Introduce el numero de dias mínimos por ciudad
1
Introduce el numero de dias máximos por ciudad
2
Introduce el mínimo número de dias del recorrido
3
Introduce el mínimo precio del viaje
10
Introduce el máximo precio del viaje
1000


//////////////////
EXTRA2
//////////////////
Inserte el numero de ciudades a generar
5
```

```
Inserte el número mínimo de ciudades a visitar
4
Inserte el numero de vuelos a generar
Tiene que ser mayor o igual al numero de ciudades
10
Inserte el numero de alojamientos a generar
Tiene que ser mayor o igual al numero de ciudades
10
Introduce el numero de dias mínimos por ciudad
1
Introduce el numero de dias máximos por ciudad
4
Introduce el mínimo número de dias del recorrido
5
Introduce el mínimo precio del viaje
100
Introduce el máximo precio del viaje
1000
////////////////////

////////////////////
EXTRA3
////////////////////
Inserte el numero de ciudades a generar
10
Inserte el número mínimo de ciudades a visitar
4
Inserte el numero de vuelos a generar
Tiene que ser mayor o igual al numero de ciudades
15
Inserte el numero de alojamientos a generar
Tiene que ser mayor o igual al numero de ciudades
15
Introduce el numero de dias mínimos por ciudad
2
Introduce el numero de dias máximos por ciudad
4
Introduce el mínimo número de dias del recorrido
10
Introduce el mínimo precio del viaje
200
Introduce el máximo precio del viaje
3000
////////////////////

////////////////////
EXTRA4
////////////////////
Inserte el numero de ciudades a generar
15
Inserte el número mínimo de ciudades a visitar
12
Inserte el numero de vuelos a generar
Tiene que ser mayor o igual al numero de ciudades
20
Inserte el numero de alojamientos a generar
Tiene que ser mayor o igual al numero de ciudades
20
```

```
Introduce el numero de dias mínimos por ciudad
2
Introduce el numero de dias máximos por ciudad
6
Introduce el mínimo número de dias del recorrido
15
Introduce el mínimo precio del viaje
400
Introduce el máximo precio del viaje
10000
////////////////////

////////////////////
EXTRA5
////////////////////
Inserte el numero de ciudades a generar
20
Inserte el número mínimo de ciudades a visitar
15
Inserte el numero de vuelos a generar
Tiene que ser mayor o igual al numero de ciudades
30
Inserte el numero de alojamientos a generar
Tiene que ser mayor o igual al numero de ciudades
24
Introduce el numero de dias mínimos por ciudad
2
Introduce el numero de dias máximos por ciudad
8
Introduce el mínimo número de dias del recorrido
30
Introduce el mínimo precio del viaje
500
Introduce el máximo precio del viaje
100000
```

7.1 Prueba 1

Input (problema)

```
(define (problem agencia_viaje)
   (:domain agencia_viaje)
   (:objects
      cg1 c1 c2 c3  - ciudad
      vg1 v1 v2 v3 v4 v5  - vuelo
      h1 h2 h3 h4 h5 - hotel
      dias1 dias2 - dias_por_ciudad
   )
   (:init
      (= (num_ciudades_escogidas) 0)
      (= (min_ciudades_a_recoger) 3 )
      (= (min_dias_por_ciudad) 1 )
      (= (max_dias_por_ciudad) 2 )
      (= (num_dias_recorrido) 0)
      (= (min_dias_recorrido) 3 )
      (= (dias_por_ciudad dias1) 1)
      (= (dias_por_ciudad dias2) 2)
      (= (min_precio_plan) 10)
      (= (max_precio_plan) 1000)
      (= (precio_plan) 0)
```

```
        (current_ciudad cg1)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
        (va_a vg1 cg1 c2)
        (va_a vg1 cg1 c3)
        (va_a v1 c3 c1 )
        (va_a v2 c1 c2 )
        (va_a v3 c2 c3 )
        (va_a v4 c3 c1 )
        (va_a v5 c1 c2 )
        (esta_en h1 c2 )
        (esta_en h2 c3 )
        (esta_en h3 c1 )
        (esta_en h4 c2 )
        (esta_en h5 c3 )
        (= (precio_hotel h1) 50)
        (= (precio_hotel h2) 35)
        (= (precio_hotel h3) 245)
        (= (precio_hotel h4) 73)
        (= (precio_hotel h5) 294)
        (= (precio_vuelo v1) 119)
        (= (precio_vuelo v2) 157)
        (= (precio_vuelo v3) 198)
        (= (precio_vuelo v4) 240)
        (= (precio_vuelo v5) 99)
        (= (precio_vuelo vg1) 0)

    )

    (:goal (and
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (precio_plan)
    )
)
```

Output obtenido

```
advancing to distance:    3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C1 VG1 H3 DIAS1
        1: ANADIR_CIUDAD C1 C2 V5 H1 DIAS1
        2: ANADIR_CIUDAD C2 C3 V3 H2 DIAS1


time spent:    0.00 seconds instantiating 26 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 11 facts and 26 actions
               0.00 seconds creating final representation with 10 relevant facts, 4 relevant fluents
               0.00 seconds computing LNF
```

```
0.00 seconds building connectivity graph
0.00 seconds searching, evaluating 31 states, to a max depth of 0
0.00 seconds total time
```

7.2 Prueba 2

Input (problema)

```
(define (problem agencia_viaje)
   (:domain agencia_viaje)
   (:objects
      cg1 c1 c2 c3 c4 c5  - ciudad
      vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10  - vuelo
      h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 - hotel
      dias1 dias2 dias3 dias4 - dias_por_ciudad
   )
   (:init
      (= (num_ciudades_escogidas) 0)
      (= (min_ciudades_a_recoger) 4 )
      (= (min_dias_por_ciudad) 1 )
      (= (max_dias_por_ciudad) 4 )
      (= (num_dias_recorrido) 0)
      (= (min_dias_recorrido) 5 )
      (= (dias_por_ciudad dias1) 1)
      (= (dias_por_ciudad dias2) 2)
      (= (dias_por_ciudad dias3) 3)
      (= (dias_por_ciudad dias4) 4)
      (= (min_precio_plan) 100)
      (= (max_precio_plan) 1000)
      (= (precio_plan) 0)
      (current_ciudad cg1)
      (ciudad_visitada cg1)
      (va_a vg1 cg1 c1)
      (va_a vg1 cg1 c2)
      (va_a vg1 cg1 c3)
      (va_a vg1 cg1 c4)
      (va_a vg1 cg1 c5)
      (va_a v1 c5 c3 )
      (va_a v2 c3 c2 )
      (va_a v3 c2 c4 )
      (va_a v4 c4 c1 )
      (va_a v5 c1 c5 )
      (va_a v6 c3 c1 )
      (va_a v7 c1 c4 )
      (va_a v8 c4 c2 )
      (va_a v9 c2 c5 )
      (va_a v10 c5 c3 )
      (esta_en h1 c5 )
      (esta_en h2 c3 )
      (esta_en h3 c1 )
      (esta_en h4 c4 )
      (esta_en h5 c2 )
      (esta_en h6 c5 )
      (esta_en h7 c3 )
      (esta_en h8 c1 )
      (esta_en h9 c4 )
      (esta_en h10 c2 )
      (= (precio_hotel h1) 55)
      (= (precio_hotel h2) 260)
      (= (precio_hotel h3) 78)
      (= (precio_hotel h4) 30)
      (= (precio_hotel h5) 236)
```

```
        (= (precio_hotel h6) 77)
        (= (precio_hotel h7) 248)
        (= (precio_hotel h8) 184)
        (= (precio_hotel h9) 91)
        (= (precio_hotel h10) 207)
        (= (precio_vuelo v1) 74)
        (= (precio_vuelo v2) 14)
        (= (precio_vuelo v3) 12)
        (= (precio_vuelo v4) 86)
        (= (precio_vuelo v5) 160)
        (= (precio_vuelo v6) 96)
        (= (precio_vuelo v7) 295)
        (= (precio_vuelo v8) 77)
        (= (precio_vuelo v9) 125)
        (= (precio_vuelo v10) 226)
        (= (precio_vuelo vg1) 0)

    )

    (:goal (and
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (precio_plan)
    )
)
```

Output obtenido

```
advancing to distance:    4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C2 VG1 H10 DIAS1
        1: ANADIR_CIUDAD C2 C4 V3 H4 DIAS1
        2: ANADIR_CIUDAD C4 C1 V4 H3 DIAS1
        3: ANADIR_CIUDAD C1 C5 V5 H1 DIAS2


time spent:    0.00 seconds instantiating 120 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 17 facts and 113 actions
               0.00 seconds creating final representation with 16 relevant facts, 4 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 72 states, to a max depth of 0
               0.00 seconds total time
```

7.3 Prueba 3

Input (problema)

```
(define (problem agencia_viaje)
    (:domain agencia_viaje)
    (:objects
        cg1 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10  - ciudad
        vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15  - vuelo
        h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 - hotel
        dias2 dias3 dias4 - dias_por_ciudad
    )
    (:init
        (= (num_ciudades_escogidas) 0)
        (= (min_ciudades_a_recoger) 7 )
        (= (min_dias_por_ciudad) 2 )
        (= (max_dias_por_ciudad) 4 )
        (= (num_dias_recorrido) 0)
        (= (min_dias_recorrido) 10 )
        (= (dias_por_ciudad dias2) 2)
        (= (dias_por_ciudad dias3) 3)
        (= (dias_por_ciudad dias4) 4)
        (= (min_precio_plan) 200)
        (= (max_precio_plan) 3000)
        (= (precio_plan) 0)
        (current_ciudad cg1)
        (ciudad_visitada cg1)
        (va_a vg1 cg1 c1)
        (va_a vg1 cg1 c2)
        (va_a vg1 cg1 c3)
        (va_a vg1 cg1 c4)
        (va_a vg1 cg1 c5)
        (va_a vg1 cg1 c6)
        (va_a vg1 cg1 c7)
        (va_a vg1 cg1 c8)
        (va_a vg1 cg1 c9)
        (va_a vg1 cg1 c10)
        (va_a v1 c4 c3 )
        (va_a v2 c3 c6 )
        (va_a v3 c6 c1 )
        (va_a v4 c1 c7 )
        (va_a v5 c7 c10 )
        (va_a v6 c10 c5 )
        (va_a v7 c5 c8 )
        (va_a v8 c8 c9 )
        (va_a v9 c9 c2 )
        (va_a v10 c2 c4 )
        (va_a v11 c10 c4 )
        (va_a v12 c4 c2 )
        (va_a v13 c2 c6 )
        (va_a v14 c6 c1 )
        (va_a v15 c1 c7 )
        (esta_en h1 c9 )
        (esta_en h2 c10 )
        (esta_en h3 c4 )
        (esta_en h4 c2 )
        (esta_en h5 c6 )
        (esta_en h6 c1 )
        (esta_en h7 c7 )
        (esta_en h8 c5 )
        (esta_en h9 c3 )
        (esta_en h10 c8 )
        (esta_en h11 c9 )
        (esta_en h12 c10 )
        (esta_en h13 c4 )
        (esta_en h14 c2 )
        (esta_en h15 c6 )
```

```
        (= (precio_hotel h1) 37)
        (= (precio_hotel h2) 221)
        (= (precio_hotel h3) 31)
        (= (precio_hotel h4) 201)
        (= (precio_hotel h5) 241)
        (= (precio_hotel h6) 87)
        (= (precio_hotel h7) 79)
        (= (precio_hotel h8) 257)
        (= (precio_hotel h9) 137)
        (= (precio_hotel h10) 44)
        (= (precio_hotel h11) 71)
        (= (precio_hotel h12) 95)
        (= (precio_hotel h13) 217)
        (= (precio_hotel h14) 298)
        (= (precio_hotel h15) 250)
        (= (precio_vuelo v1) 83)
        (= (precio_vuelo v2) 160)
        (= (precio_vuelo v3) 43)
        (= (precio_vuelo v4) 154)
        (= (precio_vuelo v5) 198)
        (= (precio_vuelo v6) 194)
        (= (precio_vuelo v7) 101)
        (= (precio_vuelo v8) 154)
        (= (precio_vuelo v9) 55)
        (= (precio_vuelo v10) 55)
        (= (precio_vuelo v11) 262)
        (= (precio_vuelo v12) 77)
        (= (precio_vuelo v13) 119)
        (= (precio_vuelo v14) 68)
        (= (precio_vuelo v15) 219)
        (= (precio_vuelo vg1) 0)

    )

    (:goal (and
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (precio_plan)
    )
)
```

Output obtenido

```
advancing to distance:    7
                          6
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C6 VG1 H5 DIAS2
```

```
        1: ANADIR_CIUDAD C6 C1 V3 H6 DIAS2
        2: ANADIR_CIUDAD C1 C7 V4 H7 DIAS2
        3: ANADIR_CIUDAD C7 C10 V5 H12 DIAS2
        4: ANADIR_CIUDAD C10 C5 V6 H8 DIAS2
        5: ANADIR_CIUDAD C5 C8 V7 H10 DIAS2
        6: ANADIR_CIUDAD C8 C9 V8 H1 DIAS2


time spent:    0.00 seconds instantiating 114 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 32 facts and 114 actions
               0.00 seconds creating final representation with 31 relevant facts, 4 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.01 seconds searching, evaluating 1566 states, to a max depth of 0
```

7.4 Prueba 4

Input (problema)

```
(define (problem agencia_viaje)
   (:domain agencia_viaje)
   (:objects
      cg1 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15  - ciudad
      vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20
          - vuelo
      h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 -
          hotel
      dias2 dias3 dias4 dias5 dias6 - dias_por_ciudad
   )
   (:init
      (= (num_ciudades_escogidas) 0)
      (= (min_ciudades_a_recoger) 12 )
      (= (min_dias_por_ciudad) 2 )
      (= (max_dias_por_ciudad) 6 )
      (= (num_dias_recorrido) 0)
      (= (min_dias_recorrido) 15 )
      (= (dias_por_ciudad dias2) 2)
      (= (dias_por_ciudad dias3) 3)
      (= (dias_por_ciudad dias4) 4)
      (= (dias_por_ciudad dias5) 5)
      (= (dias_por_ciudad dias6) 6)
      (= (min_precio_plan) 400)
      (= (max_precio_plan) 10000)
      (= (precio_plan) 0)
      (current_ciudad cg1)
      (ciudad_visitada cg1)
      (va_a vg1 cg1 c1)
      (va_a vg1 cg1 c2)
      (va_a vg1 cg1 c3)
      (va_a vg1 cg1 c4)
      (va_a vg1 cg1 c5)
      (va_a vg1 cg1 c6)
      (va_a vg1 cg1 c7)
      (va_a vg1 cg1 c8)
      (va_a vg1 cg1 c9)
      (va_a vg1 cg1 c10)
      (va_a vg1 cg1 c11)
      (va_a vg1 cg1 c12)
      (va_a vg1 cg1 c13)
      (va_a vg1 cg1 c14)
      (va_a vg1 cg1 c15)
```

```
(va_a v1 c8 c4 )
(va_a v2 c4 c1 )
(va_a v3 c1 c14 )
(va_a v4 c14 c6 )
(va_a v5 c6 c12 )
(va_a v6 c12 c11 )
(va_a v7 c11 c2 )
(va_a v8 c2 c15 )
(va_a v9 c15 c9 )
(va_a v10 c9 c7 )
(va_a v11 c7 c5 )
(va_a v12 c5 c10 )
(va_a v13 c10 c13 )
(va_a v14 c13 c3 )
(va_a v15 c3 c8 )
(va_a v16 c6 c2 )
(va_a v17 c2 c15 )
(va_a v18 c15 c7 )
(va_a v19 c7 c10 )
(va_a v20 c10 c14 )
(esta_en h1 c5 )
(esta_en h2 c6 )
(esta_en h3 c2 )
(esta_en h4 c15 )
(esta_en h5 c7 )
(esta_en h6 c10 )
(esta_en h7 c14 )
(esta_en h8 c13 )
(esta_en h9 c12 )
(esta_en h10 c8 )
(esta_en h11 c4 )
(esta_en h12 c9 )
(esta_en h13 c3 )
(esta_en h14 c1 )
(esta_en h15 c11 )
(esta_en h16 c5 )
(esta_en h17 c6 )
(esta_en h18 c2 )
(esta_en h19 c15 )
(esta_en h20 c7 )
(= (precio_hotel h1) 172)
(= (precio_hotel h2) 56)
(= (precio_hotel h3) 128)
(= (precio_hotel h4) 19)
(= (precio_hotel h5) 199)
(= (precio_hotel h6) 165)
(= (precio_hotel h7) 66)
(= (precio_hotel h8) 51)
(= (precio_hotel h9) 216)
(= (precio_hotel h10) 128)
(= (precio_hotel h11) 40)
(= (precio_hotel h12) 246)
(= (precio_hotel h13) 243)
(= (precio_hotel h14) 296)
(= (precio_hotel h15) 206)
(= (precio_hotel h16) 38)
(= (precio_hotel h17) 196)
(= (precio_hotel h18) 295)
(= (precio_hotel h19) 262)
(= (precio_hotel h20) 190)
(= (precio_vuelo v1) 97)
(= (precio_vuelo v2) 238)
(= (precio_vuelo v3) 200)
```

```
        (= (precio_vuelo v4) 77)
        (= (precio_vuelo v5) 260)
        (= (precio_vuelo v6) 30)
        (= (precio_vuelo v7) 227)
        (= (precio_vuelo v8) 207)
        (= (precio_vuelo v9) 14)
        (= (precio_vuelo v10) 165)
        (= (precio_vuelo v11) 51)
        (= (precio_vuelo v12) 271)
        (= (precio_vuelo v13) 300)
        (= (precio_vuelo v14) 139)
        (= (precio_vuelo v15) 83)
        (= (precio_vuelo v16) 231)
        (= (precio_vuelo v17) 262)
        (= (precio_vuelo v18) 177)
        (= (precio_vuelo v19) 71)
        (= (precio_vuelo v20) 223)
        (= (precio_vuelo vg1) 0)

    )

    (:goal (and
        (<= (min_precio_plan) (precio_plan))
        (>= (max_precio_plan) (precio_plan))
        (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
        (<= (min_dias_recorrido) (num_dias_recorrido))
    ))
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (precio_plan)
    )
)
```

Output obtenido

```
advancing to distance:   12
                         11
                         10
                          9
                          8
                          7
                          6
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C7 VG1 H20 DIAS2
        1: ANADIR_CIUDAD C7 C5 V11 H16 DIAS2
        2: ANADIR_CIUDAD C5 C10 V12 H6 DIAS2
        3: ANADIR_CIUDAD C10 C13 V13 H8 DIAS2
        4: ANADIR_CIUDAD C13 C3 V14 H13 DIAS2
        5: ANADIR_CIUDAD C3 C8 V15 H10 DIAS2
        6: ANADIR_CIUDAD C8 C4 V1 H11 DIAS2
        7: ANADIR_CIUDAD C4 C1 V2 H14 DIAS2
```

```
 8: ANADIR_CIUDAD C1 C14 V3 H7 DIAS2
 9: ANADIR_CIUDAD C14 C6 V4 H2 DIAS2
10: ANADIR_CIUDAD C6 C12 V5 H9 DIAS2
11: ANADIR_CIUDAD C12 C11 V6 H15 DIAS2


time spent:    0.00 seconds instantiating 240 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 47 facts and 240 actions
               0.00 seconds creating final representation with 46 relevant facts, 4 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
              10.75 seconds searching, evaluating 1727106 states, to a max depth of 0
              10.75 seconds total time
```

7.5 Prueba 5

Input (problema)

```
(define (problem agencia_viaje)
   (:domain agencia_viaje)
   (:objects
      cg1 c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18 c19 c20
          - ciudad
      vg1 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16 v17 v18 v19 v20
          v21 v22 v23 v24 v25 v26 v27 v28 v29 v30  - vuelo
      h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21
          h22 h23 h24 - hotel
      dias2 dias3 dias4 dias5 dias6 dias7 dias8 - dias_por_ciudad
   )
   (:init
      (= (num_ciudades_escogidas) 0)
      (= (min_ciudades_a_recoger) 15 )
      (= (min_dias_por_ciudad) 2 )
      (= (max_dias_por_ciudad) 8 )
      (= (num_dias_recorrido) 0)
      (= (min_dias_recorrido) 30 )
      (= (dias_por_ciudad dias2) 2)
      (= (dias_por_ciudad dias3) 3)
      (= (dias_por_ciudad dias4) 4)
      (= (dias_por_ciudad dias5) 5)
      (= (dias_por_ciudad dias6) 6)
      (= (dias_por_ciudad dias7) 7)
      (= (dias_por_ciudad dias8) 8)
      (= (min_precio_plan) 500)
      (= (max_precio_plan) 100000)
      (= (precio_plan) 0)
      (current_ciudad cg1)
      (ciudad_visitada cg1)
      (va_a vg1 cg1 c1)
      (va_a vg1 cg1 c2)
      (va_a vg1 cg1 c3)
      (va_a vg1 cg1 c4)
      (va_a vg1 cg1 c5)
      (va_a vg1 cg1 c6)
      (va_a vg1 cg1 c7)
      (va_a vg1 cg1 c8)
      (va_a vg1 cg1 c9)
      (va_a vg1 cg1 c10)
      (va_a vg1 cg1 c11)
      (va_a vg1 cg1 c12)
```

```
(va_a vg1 cg1 c13)
(va_a vg1 cg1 c14)
(va_a vg1 cg1 c15)
(va_a vg1 cg1 c16)
(va_a vg1 cg1 c17)
(va_a vg1 cg1 c18)
(va_a vg1 cg1 c19)
(va_a vg1 cg1 c20)
(va_a v1 c2 c10 )
(va_a v2 c10 c8 )
(va_a v3 c8 c5 )
(va_a v4 c5 c7 )
(va_a v5 c7 c13 )
(va_a v6 c13 c9 )
(va_a v7 c9 c18 )
(va_a v8 c18 c20 )
(va_a v9 c20 c11 )
(va_a v10 c11 c6 )
(va_a v11 c6 c19 )
(va_a v12 c19 c14 )
(va_a v13 c14 c1 )
(va_a v14 c1 c3 )
(va_a v15 c3 c15 )
(va_a v16 c15 c12 )
(va_a v17 c12 c17 )
(va_a v18 c17 c4 )
(va_a v19 c4 c16 )
(va_a v20 c16 c2 )
(va_a v21 c12 c2 )
(va_a v22 c2 c18 )
(va_a v23 c18 c19 )
(va_a v24 c19 c14 )
(va_a v25 c14 c11 )
(va_a v26 c11 c1 )
(va_a v27 c1 c3 )
(va_a v28 c3 c17 )
(va_a v29 c17 c20 )
(va_a v30 c20 c4 )
(esta_en h1 c6 )
(esta_en h2 c12 )
(esta_en h3 c2 )
(esta_en h4 c18 )
(esta_en h5 c19 )
(esta_en h6 c14 )
(esta_en h7 c11 )
(esta_en h8 c1 )
(esta_en h9 c3 )
(esta_en h10 c17 )
(esta_en h11 c20 )
(esta_en h12 c4 )
(esta_en h13 c10 )
(esta_en h14 c7 )
(esta_en h15 c8 )
(esta_en h16 c9 )
(esta_en h17 c15 )
(esta_en h18 c13 )
(esta_en h19 c16 )
(esta_en h20 c5 )
(esta_en h21 c6 )
(esta_en h22 c12 )
(esta_en h23 c2 )
(esta_en h24 c18 )
(= (precio_hotel h1) 174)
```

```
    (= (precio_hotel h2) 283)
    (= (precio_hotel h3) 64)
    (= (precio_hotel h4) 36)
    (= (precio_hotel h5) 46)
    (= (precio_hotel h6) 261)
    (= (precio_hotel h7) 144)
    (= (precio_hotel h8) 17)
    (= (precio_hotel h9) 288)
    (= (precio_hotel h10) 73)
    (= (precio_hotel h11) 282)
    (= (precio_hotel h12) 157)
    (= (precio_hotel h13) 199)
    (= (precio_hotel h14) 251)
    (= (precio_hotel h15) 106)
    (= (precio_hotel h16) 185)
    (= (precio_hotel h17) 11)
    (= (precio_hotel h18) 30)
    (= (precio_hotel h19) 35)
    (= (precio_hotel h20) 216)
    (= (precio_hotel h21) 250)
    (= (precio_hotel h22) 111)
    (= (precio_hotel h23) 251)
    (= (precio_hotel h24) 243)
    (= (precio_vuelo v1) 248)
    (= (precio_vuelo v2) 183)
    (= (precio_vuelo v3) 19)
    (= (precio_vuelo v4) 196)
    (= (precio_vuelo v5) 76)
    (= (precio_vuelo v6) 142)
    (= (precio_vuelo v7) 30)
    (= (precio_vuelo v8) 45)
    (= (precio_vuelo v9) 42)
    (= (precio_vuelo v10) 160)
    (= (precio_vuelo v11) 16)
    (= (precio_vuelo v12) 180)
    (= (precio_vuelo v13) 288)
    (= (precio_vuelo v14) 150)
    (= (precio_vuelo v15) 74)
    (= (precio_vuelo v16) 237)
    (= (precio_vuelo v17) 129)
    (= (precio_vuelo v18) 112)
    (= (precio_vuelo v19) 198)
    (= (precio_vuelo v20) 116)
    (= (precio_vuelo v21) 282)
    (= (precio_vuelo v22) 165)
    (= (precio_vuelo v23) 132)
    (= (precio_vuelo v24) 195)
    (= (precio_vuelo v25) 101)
    (= (precio_vuelo v26) 110)
    (= (precio_vuelo v27) 167)
    (= (precio_vuelo v28) 253)
    (= (precio_vuelo v29) 231)
    (= (precio_vuelo v30) 101)
    (= (precio_vuelo vg1) 0)

)

(:goal (and
    (<= (min_precio_plan) (precio_plan))
    (>= (max_precio_plan) (precio_plan))
    (<= (min_ciudades_a_recoger) (num_ciudades_escogidas))
    (<= (min_dias_recorrido) (num_dias_recorrido))
))
```

```
    ;; maximize negativo minimize negativo o viceversa
    ;; minimize va DESPUES del goal
    (:metric minimize
        (precio_plan)
    )
)
```

Output obtenido

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

ff: found legal plan as follows

step    0: ANADIR_CIUDAD CG1 C20 VG1 H11 DIAS2
        1: ANADIR_CIUDAD C20 C4 V30 H12 DIAS2
        2: ANADIR_CIUDAD C4 C16 V19 H19 DIAS2
        3: ANADIR_CIUDAD C16 C2 V20 H3 DIAS8
        4: ANADIR_CIUDAD C2 C10 V1 H13 DIAS2
        5: ANADIR_CIUDAD C10 C8 V2 H15 DIAS2
        6: ANADIR_CIUDAD C8 C5 V3 H20 DIAS2
        7: ANADIR_CIUDAD C5 C7 V4 H14 DIAS2
        8: ANADIR_CIUDAD C7 C13 V5 H18 DIAS2
        9: ANADIR_CIUDAD C13 C9 V6 H16 DIAS2
       10: ANADIR_CIUDAD C9 C18 V7 H4 DIAS2
       11: ANADIR_CIUDAD C18 C19 V23 H5 DIAS2
       12: ANADIR_CIUDAD C19 C14 V12 H6 DIAS2
       13: ANADIR_CIUDAD C14 C11 V25 H7 DIAS2
       14: ANADIR_CIUDAD C11 C1 V26 H8 DIAS2


time spent:    0.00 seconds instantiating 420 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 62 facts and 420 actions
               0.00 seconds creating final representation with 61 relevant facts, 4 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
              42.59 seconds searching, evaluating 2075793 states, to a max depth of 0
              42.59 seconds total time
```

A partir de las gráficas 1 y 2 podemos deducir que el tiempo de ejecución aumenta polinómicamente respecto al tamaño del problema en caso mejor. Es posible que la relación entre tamaño y tiempo de ejecución sea exponencial, un caso considerablemente peor, pero sería necesario realizar más experimentos para poder demostrar de qué caso se trata.

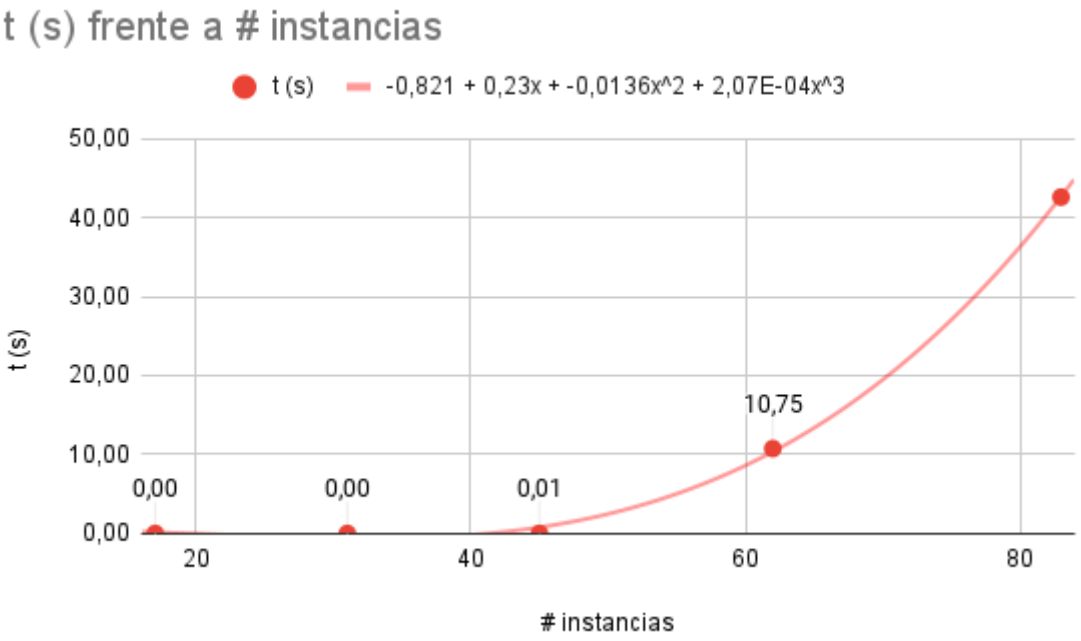

Figure 1: Tiempo al aumentar el número de instancias

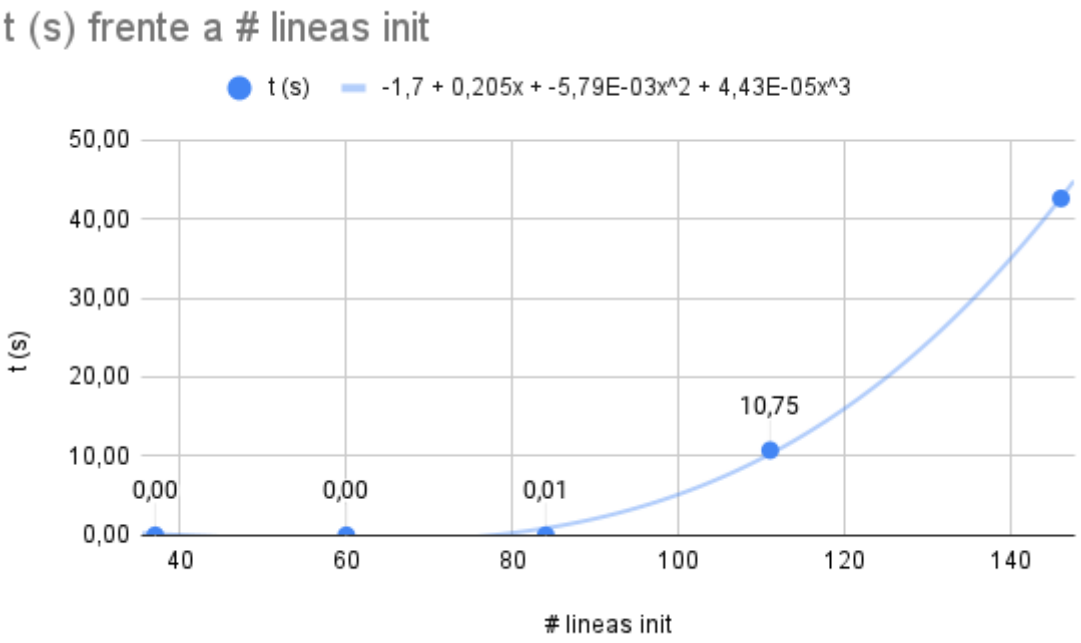

Figure 2: Tiempo al aumentar el número de líneas dde init

n ciud	min ciud	n vuelos	n aloj	n dias min	n dias max	min dias	min p	max p
3	3	5	5	1	2	3	10	1000
5	4	10	10	1	4	5	100	1000
10	4	15	15	2	4	10	200	3000
15	12	20	20	2	6	15	400	10000
20	15	30	24	2	8	30	500	100000

7.6 Observaciones

Hemos comprobado que el tiempo de ejecución aumenta mucho si los días por ciudad pueden variar mucho, ya que incrementa bastante la ramificación del operador.