Universitat Politècnica de Catalunya

Inteligencia Artificial

Bachelor Degree in Computer Science

Práctica de Planificación

Authors:
Víctor Giménez Ábalos
Guillem Ferrer Nicolas
Jordi Armengol Estapé

Profesor:
Javier Béjar Alonso

Cuadrimestre de primavera del curso 2017-2018

Contents

1 Introducción

El presente documento corresponde a la memoria de la práctica de Planificación de la asignatura de IA del segundo cuadrimestre del curso 17/18, referente al problema de las recomendaciones de viajes.

El problema a resolver es similar al de la práctica anterior, la de los SBC. La agencia de viajes Al Fin Del Mundo Y Más Allá desearía una versión más sencilla de su sistema de recomendación de viajes para su página web y se ha decidido resolverlo mediante un planificador.

A partir de ciertas restricciones del usuario y de la base de datos y hoteles de la agencia generaremos un plan de viaje.

Consideraremos cinco niveles de complejidad del problema:

- Nivel básico: partiremos del conjunto de ciudades, hoteles y vuelos y generaremos un viaje con un mínimo de ciudades a visitar.
- Extensión 1: también tendremos en cuenta un mínimo y máximo de días de las estancias y el mínimo de días totales.
- Extensión 2: todo lo anterior, pero considerando los intereses de las ciudades (un entero entre 1 y 3). Minimizaremos el interés del viaje (a menor el entero que lo representa, mejor el interés).
- Extensión 3: además de la extensión 1, consideraremos los precios de los hoteles y los minimizaremos. Se deberá respetar unos presupuestos mínimo y máximo dados.
- Extensión 4: fusionaremos las extensiones 2 y 3, ponderando el interés y el precio.

Lo consideraremos un problema de síntesis, porque no existe un conjunto acotado de posibles ciudades sino que pueden aparecer múltiples combinaciones de ciudades, trayectos y hoteles. Así, como veremos en el modelado del dominio y el problema, partiremos de un viaje inicial vacío e iremos añadiendo vuelos y estancias con las acciones.

El dominio y los diferentes problemas o casos se modelarán e implementarán con el lenguaje de planificación PDDL, con varias extensiones:

- Strips: la expresividad básica.
- ADL: aporta un lenguaje avanzado con cuantificadores, efectos condicionales.
- Typing: para tener un sistema de tipos.
- Fluents: para poder usar elementos numéricos, realizar comparaciones y operaciones aritméticas, y definir variables o funciones que además se pueden optimizar.

El programa de planificación será Fast Forward, en su versión métrica.

Así mismo, desarrollaremos un pequeño programa para generar los juegos de prueba con un tamaño creciente. También estudiaremos la evolución del tiempo de resolución en la extensión 3 con relación al número de ciudades.

Cabe decir que hemos llevado a cabo un desarrollo incremental, desarrollando cada nueva extensión partiendo del nivel anterior. En las secciones de modelado del dominio y los problemas explicaremos la versión definitiva (esto es, la extensión 4), entendiendo que incluye todas las demás, pero en la sección de desarrollo explicitaremos las iteraciones correspondientes a cada nueva extensión.

A continuación, el desarrollo de las secciones.

2 Modelado del dominio

Explicaremos como hemos modelado el dominio, que será siempre el mismo para todos los problemas (el correspondiente a cada extensión, se entiende). Se trata de los tipos, variables, predicados y acciones de la versión definitiva, pues las demás extensiones son subconjuntos de ella. Recordemos que se trata de un problema de síntesis, cosa que deberemos tener en mente al modelar la solución, sobre todo al pensar las acciones.

2.1 Tipos

Consideramos los siguientes tipos (ambos de la clase object) con nombres autoexplicativos:

- Hotel.
- City.

2.2 Variables

Usaremos las siguientes variables:

- citiesVisited: el número de ciudades visitadas. Se calculará dinámicamente.
- totalDays: el número total de días del viaje. Se indicará en el problema.
- minDaysInCity: el número mínimo de días a pasar en una misma ciudad. También se indicará en el problema.
- maxDaysInCity: lo mismo que la variable anterior, pero para el número máximo.
- stayDuration: con una ciudad como parámetro, indicará la duración de la estancia en días para cada ciudad.
- cityInterest: con una ciudad como parámetro, indicará el interés de una ciudad (medido como un entero, siendo 1 el máximo interés posible).
- totalInterest: el interés total del viaje. Se calculará dinámicamente.
- totalCost: el coste total del viaje. Se calculará dinámicamente.
- hotelCost: con un hotel como parámetro, indicará el coste por día de dicho hotel. Se indicará en el problema.
- flightCost: con dos ciudades como parámetro, indicará el coste de un vuelo de ida entre la primera y la segunda. También se indicará en el problema.

2.3 Predicados

Contamos con los siguientes predicados:

- hotelAt: con un hotel y una ciudad como parámetros, indicará si dicho hotel se encuentra en la ciudad.
- flight: con dos ciudades como parámetro, indicará la existencia de un vuelo entre ambas (solo de ida).
- currentLocation: el único parámetro será la ciudad donde se encuentre el planificador en cada momento.
- stayedAt: solamente con una ciudad como parámetro, se indicará si se ha visitado dicha ciudad.
- stayedAth: lo mismo que el anterior, pero para si nos hemos alojado en un cierto hotel.

2.4 Acciones

Las acciones del dominio serán las siguientes:

- flyAndStay: significará volar a una ciudad y llevar a cabo una estancia en ella. Como parámetros, contamos con una ciudad de la que partir, una ciudad a la que ir y un hotel en el que alojarse (de la ciudad a la que iremos). Como precondición, en el momento tendremos que encontrarnos en la ciudad de la que vamos a partir; el hotel al que alojarse deberá estar situado en la ciudad a la que viajaremos; deberá existir un vuelo (de ida) que conecte las dos ciudades; además, la ciudad a la que partir aún no la deberemos haber visitado. En cuando al efecto, como *delete list* tendremos que la localización actual dejará de ser la ciudad de la que partimos. Como *add list*, la ciudad en la que nos encontremos pasará a ser la ciudad de destino. Además, habremos visitado la ciudad de destino y realizado una estancia en el hotel en cuestión. Incrementaremos el número de ciudades visitadas en uno. También incrementaremos la duración de la estancia en la ciudad y la total en el número mínimo de días a pasar en una ciudad. Sumaremos al interés total el interés de la nueva ciudad visitada. Finalmente, sumaremos al coste total el coste por día del hotel multiplicado por el mínimo de días y el coste del vuelo.
- IncreaseStay: partiendo de la estancia mínima conseguida con flyAndStay, con IncreaseStay se añadirán días de estancia. Como parámetros, tomará una ciudad y un hotel. En la precondición, forzaremos que dicha ciudad se haya visitado, que el hotel esté situado en la ciudad, que nos hayamos alojado en el hotel y que la duración de la estancia en la ciudad sea menor al máximo de días a pasar en una misma ciudad. En cuanto al efecto, incrementaremos la estancia en la ciudad y los días totales del voaje en uno. Así mismo, incrementaremos el coste total en el coste por día del hotel (pues con esta acción estaremos añadiendo un solo día).

3 Modelado de los problemas

Procedemos a desarollar cómo hemos modelado los problemas de forma general, indicando los objetos, estado inicial y estado final que deberán indicarse en cada problema. Estos serán distintos en cada caso, porque obviamente dependerán del problema, pero seguirán una estructura común. Al tratarse de un problema de síntesis, habrá que inicializar varias condiciones a las propias del viaje vacío, como veremos a continuación.

3.1 Objetos

Como objetos del problema, indicaremos:

- El conjunto de ciudades a considerar, obviamente del tipo City, incluyendo una ciudad Origin.
- El conjunto de hoteles a tomar en cuenta, obviamente del tipo Hotel.

3.2 Estado inicial

Por otra parte, recordemos que se trata de un problema de síntesis, así que en el estado inicial deberemos generar un viaje vacío. Concretamente, indicaremos:

- Con el predicado currentLocation, que partimos de la ciudad Origin.
- Con el predicado hotelAt, que cada hotel se encuentra en su correspondiente ciudad, una vez para cada uno de los hoteles. Dependerá de la base de datos de la agencia, de cada caso.
- Con el predicado flight, todas las conexiones entre ciudades, una vez para cada vuelo existente (de ida o de vuelta). Para facilitar que el planificador pueda encontrar una solución, en los juegos de pruebas asumiremos que la ciudad Origin tendrá un vuelo de ida con todas las demás ciudades. Respecto a las demás, no asumiremos nada en particular, pero obviamente el grafo debería ser lo suficientemente connexo para que se pueda encontrar un itinerario. También dependería de la base de datos de la agencia.
- Se inicializará citiesVisited a 0, porque al principio no habremos visitado ninguna ciudad.
- Se inicializará totalDays a 0, porque al principio partiremos de un viaje vacío.
- Se inicializará minDaysInCity al mínimo de días que se quiere estar en una misma ciudad. Dependerá de las preferencias del usuario.
- Se inicializará maxDaysInCity al máximo de días que se deseen para una estancia en una misma ciudad. También dependerá de las preferencias del usuario, pero debería ser coherente con minDaysInCity.
- Se inicializará totalCost a 0, porque partiremos de un viaje vacío.

- Inicializaremos stayDuration a cero refieriendo a cada ciudad (una vez para cada ciudad, Origin incluido).
- Inicializaremos hotelCost al coste de cada hotel, una vez para cada hotel. Dependería de la base de datos de la agencia, pero hay que tener en cuenta que con precios muy elevados podría ser imposible generar una solución.
- Se inicializará flightCost de una ciudad a otra al coste en concreto (dependerá de la base de datos de la agencia, esto es, de cada problema), una vez para cada vuelo, ya sea de ida o de vuelta. Se asumirá que el coste de los vuelos desde Origin será cero. Obviamente, con costes muy elevados podría ser imposible encontrar una solución.
- Se asignará cityInterest de cada ciudad al interés de la ciudad, una vez por cada ciudad. Los valores deberían depender de la base de datos de la agencia. El interés de Origin se deberá asignar a 0.
- Se inicializará totalInterest a 0.

3.3 Estado Final

Como estado final o objetivo, tendremos la conjunción de las siguientes condiciones:

- Que citiesVisited sea mayor o igual al número mínimo de ciudades que desee visitar el usuario.
- Que totalDays sea mayor o igual al mínimo de días que desee viajar el usuario.
- Que totalCost sea superior o igual al presupuesto mínimo y inferior o igual al presupuesto máximo (los valores no deberían ser contradictorios entre ellos y deberían ser consistentes con los precios, si se quiere encontrar una solución.

3.4 Optimizaciones

Se deberá indicar la siguiente optimización:

- Minimizar la suma de totalInterest y totalCost, porque obviamente deseamos que el viaje sea lo más interesante (recordemos que el interés es mejor como menor sea el entero que lo representa) y asequible posible, con una cierta ponderación. Concretamente, utilizaremos una ponderación balanceada. Hay que tener en cuenta que el valor medio de totalInterest es igual a $1.5 \times mincities$. Por otra parte, el valormedio del coste es igual a $MeanFlightCost \times citiesVisited + MeanHotelCost \times totalDays$. Por lo tanto, multiplicaremos el interés por el coste medio y lo dividiremos entre $mincities \times 1.5$. La justificación es que el interés tiene valores muy pequeños (valor de 1 si es una ciudad muy interesante, por ejemplo) mientras que el coste está ordenes de magnitud por encima. Para intentar equipararlos hasta cierto punto para que el interés también sea tenido en cuenta, aplicamos los cálculos explicados. El valor se debe calcular antes de introducirlo, no debe ser computado por el planeador. Si bien es un valor arbitrario, pensamos que la justificación tiene sentido y que en las pruebas se dan soluciones razonables con los precios y intereses que hemos usado.

4 Desarrollo de los modelos

4.1 Nivel básico

El nivel básico obviamente coincide con nuestra primera iteración del desarrollo. Recordemos que para este nivel solamente teníamos que, dado el conjunto de ciudades, sus hoteles y los vuelos disponibles, planificar los hoteles en el que alojarse y los vuelos a tomar (respetando un mínimo de ciudades a visitar).

Si bien hemos seguido una metodología de desarrollo incremental, como la primera versión nos sirve de base para todas las demás hemos intentado ya desarrollarla de la forma lo más limpia y extensible posible (teniendo en mente que posteriormente tendríamos que llevar a cabo ampliaciones). Dicho esto, el código es bastante simple.

Para esta versión ya teníamos los tipos definitivos, City y Hotel, porque ya aparecen en el problema y de hecho son los tipos obvios. Los predicados son hotelAt, flight, currentLocation y stayedAt, explicados previamente.

Como variables, solo hay citiesVisited, de momento. Nos dimos cuenta de que de alguna forma habría que llevar la cuenta del número de ciudades visitadas para poder satisfacer la restricción del mínimo de ciudades.

Recordemos, así mismo, que estamos ante un problema de síntesis (porque hay múltiples combinaciones de soluciones, no están acotadas), así que pensaremos la solución teniendo en cuenta que tenemos que partir de un viaje vacío e irlo completando. Pensamos una sola acción para poder ir llenando el viaje: flyAndStay, para volar a una ciudad (desde otra) y alojarse en uno de sus hoteles. Como parámetros, solo incluimos las dos ciudades (la ciudad origen y la ciudad destino) y un hotel, pues en este punto realmente no teníamos necesidad de tratar con más información. La precondición solo comprobará que nos encontremos en la ciudad origen, que el vuelo (de ida) exista, que todavia no hayamos visitado la ciudad destino y que el hotel esté situado en esta última. Por otra parte, la postcondición, además de garantizar que se tome el vuelo, cambiemos de ciudad y nos alojemos en el hotel en cuestión, también va a tener que incrementar citiesVisited en uno.

¿Por qué pensamos una sola acción? Primero, porque no necesitamos más; y segundo, porque, ya que podemos, mantener una única acción simplifica la complejidad de cara al planificador.

Una vez estuvo desarrollado el dominio, creamos algunos ficheros de problema a mano, para comprobar su correcto funcionamiento. Estas pruebas todavía no serán las definitivas, porque en este punto todavía no teníamos el programa para generar problemas.

Para los ficheros de problemas, el único objetivo a marcar en el nivel básico es que se satisfaga el número mínimo de ciudades a visitar. En la inicialización, solamente tendremos que indicar la situación de los hoteles, la situación inicial y los vuelos existentes, además de inicializar citiesVisites a 0.

4.2 Extensión 1

Basándonos en el nivel básico, teníamos que considerar que esta vez también habría mínimo y máximo de días de las estancias y un mínimo de días totales.

Nos dimos cuenta de que la extensión encajaba perfectamente con el código ya programado para el nivel 1. Básicamente, añadimos una acción para alargar las estancias en las ciudades en un día IncreaseStay. La acción simplemente toma una ciudad como parámetro, comprueba que se trate de una de las ciudades visitadas y que aún no hayamos estado más días del máximo por ciudad y entonces incrementa la duración de la estancia en la ciudad y la total. Así, para poder comprobar las nuevas restricciones, tendremos que añadir hasta 4 variables:

- totalDays.
- minDaysInCity.
- maxDaysInCity.
- stayDuration ?c - city.

Los predicados no varían en esta versión. La acción flyStay simplemente tiene que adaptarse para tener en cuenta las restricciones del número de días.

En esta versión, ya tendríamos las dos acciones de la versión final, aunque las iríamos ampliando en las siguientes extensiones.

Es en esta iteración que comenzamos el desarrollo del programa para generar ficheros de problema, y partir de aquí lo programaríamos de forma paralela a las extensiones. Esto es, en cada iteración iremos actualizando el generador para que también pueda generar problemas para las nuevas extensiones, manteniendo la compatibilidad con las antiguas.

En los ficheros de problemas, además de lo indicado en el nivel básico, para esta extensión es necesario indicar el mínimo de días totales del viaje e inicializar las variables para llevar la cuenta del número de días a 0. Hay que inicializar las variables minDaysInCity y maxDaysInCity según las preferencias de usuario.

4.3 Extensión 2

Para esta versión partimos del código de la extensión 1, pero hay que añadir el interés de las ciudades (represetado como un entero), que además se deberá minimizar. Los predicados no variarán. Sin embargo, tendremos que añadir dos variables para representar el interés: cityInterest, el interés de cada ciudad, y totalInterest, que será una variable contador y variará dinámicamente.

Las acciones son prácticamente iguales a la anterior versión. El único cambio es que flyAndSay en esta extensión también tiene que sumar el interés de las ciudades. La explicación de que el cambio sea tan trivial es que de la tarea de minimizar se encarga el propio planeador. En los ficheros de problema, además del interés de cada ciudad, hay que indicar que deseamos minimizar el interés total. Es la primera vez que indicamos una optimización, con la palabra clave metric.

Una vez adaptado el generador de problemas y probada la extensión, pasamos a la siguiente extensión.

4.4 Extensión 3

Para este nivel en lugar de partir del immediatamente anterior tenemos que basarnos nuevamente en la extensión 1. Tenemos que incorporar los costes de los hoteles y vuelos, y minimizarlos.

Como variables, fue necesario añadir variables para almacenar los costes. Unas serían fijas (representa que proporcionadas por la base de datos de la agencia):

- hotelCost ?h - hotel: para indicar el coste por noche de un hotel.
- flightcost ?from - city ?to - city: para indicar el coste de un vuelo (solo ida).

Y una actualizada dinámicamente: totalCost, para llevar la cuenta del coste total del viaje.

La acción flyAndStay, además de todo lo anterior, también tiene que añadir el coste de los vuelos. Sin embargo, respecto al coste del hotel solo puede añadir el de un día, porque es la acción IncreaseStay la que se encarga de aumentar los días de alojamiento. Así, esta última acción también debe sumar el coste de un día (por cada vez que añada uno). Por supuesto, hay que consultar el coste del hotel o vuelo en cuestión. Por otra parte, cabe decir que seguíamos contando con solo estas dos acciones.

Extendiendo el código, caímos en la cuenta de que era imperativo añadir un nuevo predicado, stayedAth, que indica si nos hemos alojado en un cierto hotel. Es necesario para que IncreaseStay sepa el coste del hotel al aumentar en un día la estancia.

El mínimo y máximo de presupuesto no se indican en el dominio, sino obviamente en el fichero de problema, como condiciones del objetivo. Además, se indica que hay que minimizar el presupuesto. Hay que recordar que hay que inicializar totalCost a 0, y los costes de cada hotel y vuelo según la base de datos de la agencia.

En cuanto al programa generador de problemas, cabe destacar que es en esta iteración en la que nos dimos cuenta de que sería necesaria la capacidad de crear problemas de distinto tamaño. Además de adaptar el programa a la extensión 3, también desarrollamos esta característica (para todas las versiones). Sin embargo, el estudio lo llevamos a cabo al final, una vez terminadas todas las extensiones.

4.5 Extensión 4

Para la última iteración, en cuanto al dominio esencialmente hemos tenido que fusionar el código de la versión 2 y la versión 3 para que tuviera en cuenta tanto los costes como los intereses, así que no ha habido muchas complicaciones.

Sin embargo, para los problemas, además de ajuntar trivialmente los elementos de las extensiones 2 y 3, se debe añadir una cierta ponderación porque el interés y el coste (ambos a minimizar) están en unidades diferentes. La ponderación usada ya ha sido explicada en el apartado del modelado de problemas.

5 Pruebas y ejecución

5.1 Script

Hemos desarrollado un pequeño programa en C++ que nos servirá para crear ficheros de problemas concretos que usaremos para las pruebas. Nos servirá para las distintas versiones de nuestra solución. De hecho, usaremos los mismos problemas con las ampliaciones debidas según cada dominio en las primeras extensiones, pero a partir de la 3 tendremos que reducir su complejidad porque los archivos de problemas de más de 200 líneas no funcionan con este planeador.

Como parámetro, el programa recibirá la extensión para la que tiene que generar problemas. Además, el programa leerá el tamaño de los distintos elementos del problema:

- numberOfCities.
- numberOfHotelsByCity.
- numberOfFlights(outdegreebycity, ¡ cities), el número de vuelos de salida por ciudad.
- numberOfCitiesToVisit.
- minDaysinCity.
- maxDaysinCity.
- minTotalDays.
- minCost.
- maxCost.

Obviamente, solo preguntará por los parámetros correspondientes a cada extensión.

El código fuente del programa generador de problemas está adjunto a la presente entrega.

Como hemos generado todas las pruebas automáticamente, a diferencia de las práctica anterior (la de los SBC) no hemos ido probando caso por caso distintas clases de equivalencia. Simplemente, hemos probado varios casos con tamaños distintos (pero, a nuestro entender, suficientemente elevados), con el límite de líneas de ficheros y que fuese viable encontrar una solución en un tiempo razonable. Hemos observado los resultados para comprobar que tenían sentido. Cabe decir que el programa en sí mismo ya incorpora una cierta aleatoriedad (por ejemplo, para escoger qué vuelos de salida tendrá cada ciudad, o para escoger los precios dentro de un rango), y es de esta forma como hemos validado que el programa funcionaba para distintas casuísticas.

5.2 Pruebas del nivel básico

5.2.1 Prueba I

Generaremos automáticamente un problema con 15 ciudades, 3 hoteles por ciudad, 3 vuelos de salida por ciudad, 12 ciudades a visitar, un mínimo de 2 días para cada estancia y un máximo

de 4. Consideramos que una prueba de este calibre es suficientemente compleja. 3 hoteles por ciudad nos permitirá probar una cierta variancia en cuanto al número de hoteles, pero a cambio habrá solamente 3 vuelos de salida por ciudad.

Si observamos la salida, comprobamos que rápidamente se nos proporciona una solución que cumple las restricciones y consiste en un plan de vuelos y estancias correcto.

Entrada:

```
(define (problem autogen)
  (:domain worldsEndAndBeyond)

  (:objects
    Origin cityA cityB cityC cityD cityE cityF cityG cityH cityI cityJ cityK
                                                cityL cityM cityN cityO - city
    hotelA_A hotelA_B hotelA_C hotelB_A hotelB_B hotelB_C hotelC_A hotelC_B
        hotelC_C hotelD_A hotelD_B hotelD_C hotelE_A hotelE_B hotelE_C hotelF_A
        hotelF_B hotelF_C hotelG_A hotelG_B hotelG_C hotelH_A hotelH_B hotelH_C
        hotelI_A hotelI_B hotelI_C hotelJ_A hotelJ_B hotelJ_C hotelK_A hotelK_B
        hotelK_C hotelL_A hotelL_B hotelL_C hotelM_A hotelM_B hotelM_C hotelN_A
                            hotelN_B hotelN_C hotelO_A hotelO_B hotelO_C - hotel
  )

  (:init
    (currentLocation Origin)

    (hotelAt hotelA_A cityA)
    (hotelAt hotelA_B cityA)
    (hotelAt hotelA_C cityA)
    (hotelAt hotelB_A cityB)
    (hotelAt hotelB_B cityB)
    (hotelAt hotelB_C cityB)
    (hotelAt hotelC_A cityC)
    (hotelAt hotelC_B cityC)
    (hotelAt hotelC_C cityC)
    (hotelAt hotelD_A cityD)
    (hotelAt hotelD_B cityD)
    (hotelAt hotelD_C cityD)
    (hotelAt hotelE_A cityE)
    (hotelAt hotelE_B cityE)
    (hotelAt hotelE_C cityE)
    (hotelAt hotelF_A cityF)
    (hotelAt hotelF_B cityF)
    (hotelAt hotelF_C cityF)
    (hotelAt hotelG_A cityG)
    (hotelAt hotelG_B cityG)
    (hotelAt hotelG_C cityG)
    (hotelAt hotelH_A cityH)
```

```
(hotelAt hotelH_B cityH)
(hotelAt hotelH_C cityH)
(hotelAt hotelI_A cityI)
(hotelAt hotelI_B cityI)
(hotelAt hotelI_C cityI)
(hotelAt hotelJ_A cityJ)
(hotelAt hotelJ_B cityJ)
(hotelAt hotelJ_C cityJ)
(hotelAt hotelK_A cityK)
(hotelAt hotelK_B cityK)
(hotelAt hotelK_C cityK)
(hotelAt hotelL_A cityL)
(hotelAt hotelL_B cityL)
(hotelAt hotelL_C cityL)
(hotelAt hotelM_A cityM)
(hotelAt hotelM_B cityM)
(hotelAt hotelM_C cityM)
(hotelAt hotelN_A cityN)
(hotelAt hotelN_B cityN)
(hotelAt hotelN_C cityN)
(hotelAt hotelO_A cityO)
(hotelAt hotelO_B cityO)
(hotelAt hotelO_C cityO)

(flight Origin cityA)
(flight Origin cityB)
(flight Origin cityC)
(flight Origin cityD)
(flight Origin cityE)
(flight Origin cityF)
(flight Origin cityG)
(flight Origin cityH)
(flight Origin cityI)
(flight Origin cityJ)
(flight Origin cityK)
(flight Origin cityL)
(flight Origin cityM)
(flight Origin cityN)
(flight Origin cityO)
(flight cityA cityC)
(flight cityA cityF)
(flight cityA cityN)
(flight cityB cityC)
(flight cityB cityF)
(flight cityB cityL)
(flight cityC cityA)
(flight cityC cityJ)
```

```
  (flight cityC cityM)
  (flight cityD cityC)
  (flight cityD cityE)
  (flight cityD cityJ)
  (flight cityE cityD)
  (flight cityE cityH)
  (flight cityE cityI)
  (flight cityF cityE)
  (flight cityF cityI)
  (flight cityF cityJ)
  (flight cityG cityB)
  (flight cityG cityK)
  (flight cityG cityO)
  (flight cityH cityD)
  (flight cityH cityL)
  (flight cityH cityM)
  (flight cityI cityA)
  (flight cityI cityC)
  (flight cityI cityM)
  (flight cityJ cityI)
  (flight cityJ cityL)
  (flight cityJ cityO)
  (flight cityK cityA)
  (flight cityK cityD)
  (flight cityK cityO)
  (flight cityL cityF)
  (flight cityL cityJ)
  (flight cityL cityK)
  (flight cityM cityA)
  (flight cityM cityC)
  (flight cityM cityL)
  (flight cityN cityB)
  (flight cityN cityD)
  (flight cityN cityL)
  (flight cityO cityD)
  (flight cityO cityJ)
  (flight cityO cityM)

  (= (citiesVisited) 0)

)

(:goal
  (>= (citiesVisited) 12)
)
```

)

Salida:

```
ff: parsing domain file
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

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

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

ff: found legal plan as follows

step    0: FLYANDSTAY ORIGIN CITYO HOTELO_A
        1: FLYANDSTAY CITYO CITYM HOTELM_A
        2: FLYANDSTAY CITYM CITYL HOTELL_A
        3: FLYANDSTAY CITYL CITYK HOTELK_A
        4: FLYANDSTAY CITYK CITYD HOTELD_A
        5: FLYANDSTAY CITYD CITYJ HOTELJ_A
        6: FLYANDSTAY CITYJ CITYI HOTELI_A
        7: FLYANDSTAY CITYI CITYC HOTELC_A
        8: FLYANDSTAY CITYC CITYA HOTELA_A
        9: FLYANDSTAY CITYA CITYN HOTELN_A
       10: FLYANDSTAY CITYN CITYB HOTELB_A
```

```
       11: FLYANDSTAY CITYB CITYF HOTELF_A


time spent:    0.00 seconds instantiating 180 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 47 facts and 180 actions
               0.00 seconds creating final representation with 46 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
```

5.2.2 Prueba II

Comentaríamos básicamente lo mismo que en la Prueba I, pero con la diferencia de que esta vez la prueba tiene un tamaño más reducido (para probarlo), con menos hoteles por ciudad pero a cambio con más posibles vuelos de salida por ciudad. Si observamos la salida, también validamos el funcionamiento del programa, aunque en este caso la solución es más pequeña.

Entrada:

```
(define (problem autogen)
  (:domain worldsEndAndBeyond)

  (:objects
    Origin cityA cityB cityC cityD cityE cityF cityG cityH cityI cityJ cityK
                                                              cityL - city
    hotelA_A hotelA_B hotelB_A hotelB_B hotelC_A hotelC_B hotelD_A
        hotelD_B hotelE_A hotelE_B hotelF_A hotelF_B hotelG_A hotelG_B
        hotelH_A hotelH_B hotelI_A hotelI_B hotelJ_A hotelJ_B hotelK_A
                                    hotelK_B hotelL_A hotelL_B - hotel
  )

  (:init
    (currentLocation Origin)

    (hotelAt hotelA_A cityA)
    (hotelAt hotelA_B cityA)
    (hotelAt hotelB_A cityB)
    (hotelAt hotelB_B cityB)
    (hotelAt hotelC_A cityC)
    (hotelAt hotelC_B cityC)
    (hotelAt hotelD_A cityD)
    (hotelAt hotelD_B cityD)
    (hotelAt hotelE_A cityE)
    (hotelAt hotelE_B cityE)
    (hotelAt hotelF_A cityF)
```

```
(hotelAt hotelF_B cityF)
(hotelAt hotelG_A cityG)
(hotelAt hotelG_B cityG)
(hotelAt hotelH_A cityH)
(hotelAt hotelH_B cityH)
(hotelAt hotelI_A cityI)
(hotelAt hotelI_B cityI)
(hotelAt hotelJ_A cityJ)
(hotelAt hotelJ_B cityJ)
(hotelAt hotelK_A cityK)
(hotelAt hotelK_B cityK)
(hotelAt hotelL_A cityL)
(hotelAt hotelL_B cityL)

(flight Origin cityA)
(flight Origin cityB)
(flight Origin cityC)
(flight Origin cityD)
(flight Origin cityE)
(flight Origin cityF)
(flight Origin cityG)
(flight Origin cityH)
(flight Origin cityI)
(flight Origin cityJ)
(flight Origin cityK)
(flight Origin cityL)
(flight cityA cityC)
(flight cityA cityE)
(flight cityA cityF)
(flight cityA cityH)
(flight cityB cityF)
(flight cityB cityH)
(flight cityB cityI)
(flight cityB cityJ)
(flight cityC cityF)
(flight cityC cityG)
(flight cityC cityH)
(flight cityC cityK)
(flight cityD cityC)
(flight cityD cityH)
(flight cityD cityK)
(flight cityD cityL)
(flight cityE cityD)
(flight cityE cityF)
(flight cityE cityH)
(flight cityE cityL)
(flight cityF cityC)
```

```
    (flight cityF cityD)
    (flight cityF cityH)
    (flight cityF cityJ)
    (flight cityG cityA)
    (flight cityG cityE)
    (flight cityG cityJ)
    (flight cityG cityK)
    (flight cityH cityC)
    (flight cityH cityE)
    (flight cityH cityG)
    (flight cityH cityI)
    (flight cityI cityB)
    (flight cityI cityC)
    (flight cityI cityD)
    (flight cityI cityH)
    (flight cityJ cityA)
    (flight cityJ cityC)
    (flight cityJ cityE)
    (flight cityJ cityG)
    (flight cityK cityC)
    (flight cityK cityD)
    (flight cityK cityH)
    (flight cityK cityJ)
    (flight cityL cityA)
    (flight cityL cityE)
    (flight cityL cityI)
    (flight cityL cityJ)

    (= (citiesVisited) 0)

  )

  (:goal
    (>= (citiesVisited) 6)
  )


)
```

Salida:

```
ff: parsing domain file
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
```

```
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

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

advancing to distance:    6
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: FLYANDSTAY ORIGIN CITYL HOTELL_A
        1: FLYANDSTAY CITYL CITYJ HOTELJ_A
        2: FLYANDSTAY CITYJ CITYG HOTELG_A
        3: FLYANDSTAY CITYG CITYK HOTELK_A
        4: FLYANDSTAY CITYK CITYH HOTELH_A
        5: FLYANDSTAY CITYH CITYI HOTELI_A


time spent:    0.00 seconds instantiating 120 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 38 facts and 120 actions
               0.00 seconds creating final representation with 37 relevant facts,
                                                          1 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 30 states, to a max depth of 0
               0.00 seconds total time
```

5.3 Pruebas de la extensión 1

En este caso, utilizamos las mismas dos pruebas que en la base, pero añadiendo los días totales y por estancia (y sus respectivas restricciones).

Para la Prueba I, el mínimo de días en una misma ciudad será 2, el máximo será 4 y el mínimo total de días será. En cambio, en la Prueba II el rango de días por ciudad será entre 1 y 2.

Los resultados son aparentemente correctos (observamos que se cumplen todas las restricciones). En las soluciones ya aparece IncreaseStay. Sin embargo, continúa siendo relativamente fácil y rápido para el planificador.

5.3.1 Prueba I

Entrada:

```
(define (problem autogen)
  (:domain worldsEndAndBeyond)

  (:objects
    Origin cityA cityB cityC cityD cityE cityF cityG cityH cityI cityJ cityK
                                                cityL cityM cityN cityO - city
    hotelA_A hotelA_B hotelA_C hotelB_A hotelB_B hotelB_C hotelC_A
        hotelC_B hotelC_C hotelD_A hotelD_B hotelD_C hotelE_A hotelE_B
        hotelE_C hotelF_A hotelF_B hotelF_C hotelG_A hotelG_B hotelG_C
        hotelH_A hotelH_B hotelH_C hotelI_A hotelI_B hotelI_C hotelJ_A
        hotelJ_B hotelJ_C hotelK_A hotelK_B hotelK_C hotelL_A hotelL_B
        hotelL_C hotelM_A hotelM_B hotelM_C hotelN_A hotelN_B hotelN_C
                                    hotelO_A hotelO_B hotelO_C - hotel
  )

  (:init
    (currentLocation Origin)

    (hotelAt hotelA_A cityA)
    (hotelAt hotelA_B cityA)
    (hotelAt hotelA_C cityA)
    (hotelAt hotelB_A cityB)
    (hotelAt hotelB_B cityB)
    (hotelAt hotelB_C cityB)
    (hotelAt hotelC_A cityC)
    (hotelAt hotelC_B cityC)
    (hotelAt hotelC_C cityC)
    (hotelAt hotelD_A cityD)
    (hotelAt hotelD_B cityD)
    (hotelAt hotelD_C cityD)
    (hotelAt hotelE_A cityE)
    (hotelAt hotelE_B cityE)
    (hotelAt hotelE_C cityE)
    (hotelAt hotelF_A cityF)
    (hotelAt hotelF_B cityF)
    (hotelAt hotelF_C cityF)
    (hotelAt hotelG_A cityG)
    (hotelAt hotelG_B cityG)
```

```
(hotelAt hotelG_C cityG)
(hotelAt hotelH_A cityH)
(hotelAt hotelH_B cityH)
(hotelAt hotelH_C cityH)
(hotelAt hotelI_A cityI)
(hotelAt hotelI_B cityI)
(hotelAt hotelI_C cityI)
(hotelAt hotelJ_A cityJ)
(hotelAt hotelJ_B cityJ)
(hotelAt hotelJ_C cityJ)
(hotelAt hotelK_A cityK)
(hotelAt hotelK_B cityK)
(hotelAt hotelK_C cityK)
(hotelAt hotelL_A cityL)
(hotelAt hotelL_B cityL)
(hotelAt hotelL_C cityL)
(hotelAt hotelM_A cityM)
(hotelAt hotelM_B cityM)
(hotelAt hotelM_C cityM)
(hotelAt hotelN_A cityN)
(hotelAt hotelN_B cityN)
(hotelAt hotelN_C cityN)
(hotelAt hotelO_A cityO)
(hotelAt hotelO_B cityO)
(hotelAt hotelO_C cityO)

(flight Origin cityA)
(flight Origin cityB)
(flight Origin cityC)
(flight Origin cityD)
(flight Origin cityE)
(flight Origin cityF)
(flight Origin cityG)
(flight Origin cityH)
(flight Origin cityI)
(flight Origin cityJ)
(flight Origin cityK)
(flight Origin cityL)
(flight Origin cityM)
(flight Origin cityN)
(flight Origin cityO)
(flight cityA cityC)
(flight cityA cityD)
(flight cityA cityI)
(flight cityB cityF)
(flight cityB cityI)
(flight cityB cityJ)
```

```
(flight cityC cityB)
(flight cityC cityD)
(flight cityC cityK)
(flight cityD cityA)
(flight cityD cityB)
(flight cityD cityJ)
(flight cityE cityH)
(flight cityE cityM)
(flight cityE cityO)
(flight cityF cityI)
(flight cityF cityM)
(flight cityF cityO)
(flight cityG cityA)
(flight cityG cityN)
(flight cityG cityO)
(flight cityH cityG)
(flight cityH cityK)
(flight cityH cityM)
(flight cityI cityA)
(flight cityI cityC)
(flight cityI cityN)
(flight cityJ cityE)
(flight cityJ cityN)
(flight cityJ cityO)
(flight cityK cityC)
(flight cityK cityD)
(flight cityK cityI)
(flight cityL cityM)
(flight cityL cityN)
(flight cityL cityO)
(flight cityM cityD)
(flight cityM cityH)
(flight cityM cityN)
(flight cityN cityB)
(flight cityN cityF)
(flight cityN cityL)
(flight cityO cityC)
(flight cityO cityL)
(flight cityO cityM)

(= (citiesVisited) 0)
(= (totalDays) 0)
(= (minDaysInCity) 2)
(= (maxDaysInCity) 4)

(= (stayDuration Origin) 0)
(= (stayDuration cityA) 0)
```

```
    (= (stayDuration cityB) 0)
    (= (stayDuration cityC) 0)
    (= (stayDuration cityD) 0)
    (= (stayDuration cityE) 0)
    (= (stayDuration cityF) 0)
    (= (stayDuration cityG) 0)
    (= (stayDuration cityH) 0)
    (= (stayDuration cityI) 0)
    (= (stayDuration cityJ) 0)
    (= (stayDuration cityK) 0)
    (= (stayDuration cityL) 0)
    (= (stayDuration cityM) 0)
    (= (stayDuration cityN) 0)
    (= (stayDuration cityO) 0)

  )

  (:goal
    (and
      (>= (citiesVisited) 12)
      (>= (totalDays) 20)
    )
  )


)
```

Salida:

```
ff: parsing domain file
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

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

advancing to distance:   12
                         11
```

```
                        10
                         9
                         8
                         7
                         6
                         5
                         4
                         3
                         2
                         1
                         0

ff: found legal plan as follows

step    0: FLYANDSTAY ORIGIN CITYO HOTELO_A
        1: FLYANDSTAY CITYO CITYM HOTELM_A
        2: FLYANDSTAY CITYM CITYN HOTELN_A
        3: FLYANDSTAY CITYN CITYB HOTELB_A
        4: FLYANDSTAY CITYB CITYI HOTELI_A
        5: FLYANDSTAY CITYI CITYC HOTELC_A
        6: FLYANDSTAY CITYC CITYK HOTELK_A
        7: FLYANDSTAY CITYK CITYD HOTELD_A
        8: FLYANDSTAY CITYD CITYJ HOTELJ_A
        9: FLYANDSTAY CITYJ CITYE HOTELE_A
       10: FLYANDSTAY CITYE CITYH HOTELH_A
       11: FLYANDSTAY CITYH CITYG HOTELG_A


time spent:    0.00 seconds instantiating 196 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 47 facts and 195 actions
               0.00 seconds creating final representation with 46 relevant facts,
                                                              33 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 108 states, to a max depth of 0
               0.00 seconds total time
```

5.3.2 Prueba I

Entrada:

```
(define (problem autogen)
  (:domain worldsEndAndBeyond)
```

```
(:objects
  Origin cityA cityB cityC cityD cityE cityF cityG cityH cityI cityJ cityK
                                                              cityL - city
  hotelA_A hotelA_B hotelB_A hotelB_B hotelC_A hotelC_B hotelD_A
      hotelD_B hotelE_A hotelE_B hotelF_A hotelF_B hotelG_A hotelG_B
      hotelH_A hotelH_B hotelI_A hotelI_B hotelJ_A hotelJ_B hotelK_A
                                  hotelK_B hotelL_A hotelL_B - hotel
)

(:init
  (currentLocation Origin)

  (hotelAt hotelA_A cityA)
  (hotelAt hotelA_B cityA)
  (hotelAt hotelB_A cityB)
  (hotelAt hotelB_B cityB)
  (hotelAt hotelC_A cityC)
  (hotelAt hotelC_B cityC)
  (hotelAt hotelD_A cityD)
  (hotelAt hotelD_B cityD)
  (hotelAt hotelE_A cityE)
  (hotelAt hotelE_B cityE)
  (hotelAt hotelF_A cityF)
  (hotelAt hotelF_B cityF)
  (hotelAt hotelG_A cityG)
  (hotelAt hotelG_B cityG)
  (hotelAt hotelH_A cityH)
  (hotelAt hotelH_B cityH)
  (hotelAt hotelI_A cityI)
  (hotelAt hotelI_B cityI)
  (hotelAt hotelJ_A cityJ)
  (hotelAt hotelJ_B cityJ)
  (hotelAt hotelK_A cityK)
  (hotelAt hotelK_B cityK)
  (hotelAt hotelL_A cityL)
  (hotelAt hotelL_B cityL)

  (flight Origin cityA)
  (flight Origin cityB)
  (flight Origin cityC)
  (flight Origin cityD)
  (flight Origin cityE)
  (flight Origin cityF)
  (flight Origin cityG)
  (flight Origin cityH)
  (flight Origin cityI)
  (flight Origin cityJ)
```

```
(flight Origin cityK)
(flight Origin cityL)
(flight cityA cityB)
(flight cityA cityD)
(flight cityA cityI)
(flight cityA cityL)
(flight cityB cityE)
(flight cityB cityG)
(flight cityB cityJ)
(flight cityB cityL)
(flight cityC cityA)
(flight cityC cityB)
(flight cityC cityJ)
(flight cityC cityL)
(flight cityD cityA)
(flight cityD cityH)
(flight cityD cityI)
(flight cityD cityL)
(flight cityE cityC)
(flight cityE cityD)
(flight cityE cityG)
(flight cityE cityH)
(flight cityF cityC)
(flight cityF cityG)
(flight cityF cityH)
(flight cityF cityL)
(flight cityG cityE)
(flight cityG cityF)
(flight cityG cityH)
(flight cityG cityL)
(flight cityH cityE)
(flight cityH cityF)
(flight cityH cityG)
(flight cityH cityL)
(flight cityI cityA)
(flight cityI cityB)
(flight cityI cityC)
(flight cityI cityD)
(flight cityJ cityC)
(flight cityJ cityF)
(flight cityJ cityG)
(flight cityJ cityI)
(flight cityK cityA)
(flight cityK cityF)
(flight cityK cityJ)
(flight cityK cityL)
(flight cityL cityA)
```

```
    (flight cityL cityB)
    (flight cityL cityC)
    (flight cityL cityI)

    (= (citiesVisited) 0)
    (= (totalDays) 0)
    (= (minDaysInCity) 1)
    (= (maxDaysInCity) 2)

    (= (stayDuration Origin) 0)
    (= (stayDuration cityA) 0)
    (= (stayDuration cityB) 0)
    (= (stayDuration cityC) 0)
    (= (stayDuration cityD) 0)
    (= (stayDuration cityE) 0)
    (= (stayDuration cityF) 0)
    (= (stayDuration cityG) 0)
    (= (stayDuration cityH) 0)
    (= (stayDuration cityI) 0)
    (= (stayDuration cityJ) 0)
    (= (stayDuration cityK) 0)
    (= (stayDuration cityL) 0)

  )

  (:goal
    (and
      (>= (citiesVisited) 6)
      (>= (totalDays) 20)
    )
  )


)
```

Salida:

```
ff: parsing domain file
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
 ... done.


no metric specified. plan length assumed.
```

```
checking for cyclic := effects --- OK.

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

advancing to distance:   20
                         19
                         18
                         17
                         16
                         15
                         14
                         13
                         12
                         11
                         10
                          9
                          8
                          7
                          6
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: FLYANDSTAY ORIGIN CITYL HOTELL_A
        1: INCREASESTAY CITYL
        2: FLYANDSTAY CITYL CITYI HOTELI_A
        3: INCREASESTAY CITYI
        4: FLYANDSTAY CITYI CITYD HOTELD_A
        5: INCREASESTAY CITYD
        6: FLYANDSTAY CITYD CITYH HOTELH_A
        7: INCREASESTAY CITYH
        8: FLYANDSTAY CITYH CITYG HOTELG_A
        9: INCREASESTAY CITYG
       10: FLYANDSTAY CITYG CITYF HOTELF_A
       11: INCREASESTAY CITYF
       12: FLYANDSTAY CITYF CITYC HOTELC_A
       13: INCREASESTAY CITYC
       14: FLYANDSTAY CITYC CITYA HOTELA_A
       15: INCREASESTAY CITYA
       16: FLYANDSTAY CITYA CITYB HOTELB_A
       17: INCREASESTAY CITYB
```

```
       18: FLYANDSTAY CITYB CITYJ HOTELJ_A
       19: INCREASESTAY CITYJ


time spent:    0.00 seconds instantiating 133 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 38 facts and 132 actions
               0.00 seconds creating final representation with 37 relevant facts,
                                                              27 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 79 states, to a max depth of 0
               0.00 seconds total time
```

5.4 Pruebas de la extensión 2

Igual que en la extensión 1, las pruebas siguen siendo básicamente las mismas (en cuanto a tamaño y restricciones, recordemos que hay elementos que son aleatorios, como por ejemplo el interés, dentro de un rango), pero añadiendo el interés. Por primera vez, aparece una optimización, que no consigue un óptimo global al ser greedy. Así, las soluciones nos parecen correctas en el sentido de que claramente cumplen las restricciones, pero no podemos asegurar, en absoluto, que sea el mejor interés posible.

5.4.1 Prueba I

Entrada:

```
(define (problem autogen)
  (:domain worldsEndAndBeyond)

  (:objects
    Origin cityA cityB cityC cityD cityE cityF cityG cityH cityI cityJ cityK
                                              cityL cityM cityN cityO - city
    hotelA_A hotelA_B hotelA_C hotelB_A hotelB_B hotelB_C hotelC_A hotelC_B
             hotelC_C hotelD_A hotelD_B hotelD_C hotelE_A hotelE_B hotelE_C
             hotelF_A hotelF_B hotelF_C hotelG_A hotelG_B hotelG_C hotelH_A
             hotelH_B hotelH_C hotelI_A hotelI_B hotelI_C hotelJ_A hotelJ_B
             hotelJ_C hotelK_A hotelK_B hotelK_C hotelL_A hotelL_B hotelL_C
             hotelM_A hotelM_B hotelM_C hotelN_A hotelN_B hotelN_C hotelO_A
                                                 hotelO_B hotelO_C - hotel
  )

  (:init
    (currentLocation Origin)
```

```
(hotelAt hotelA_A cityA)
(hotelAt hotelA_B cityA)
(hotelAt hotelA_C cityA)
(hotelAt hotelB_A cityB)
(hotelAt hotelB_B cityB)
(hotelAt hotelB_C cityB)
(hotelAt hotelC_A cityC)
(hotelAt hotelC_B cityC)
(hotelAt hotelC_C cityC)
(hotelAt hotelD_A cityD)
(hotelAt hotelD_B cityD)
(hotelAt hotelD_C cityD)
(hotelAt hotelE_A cityE)
(hotelAt hotelE_B cityE)
(hotelAt hotelE_C cityE)
(hotelAt hotelF_A cityF)
(hotelAt hotelF_B cityF)
(hotelAt hotelF_C cityF)
(hotelAt hotelG_A cityG)
(hotelAt hotelG_B cityG)
(hotelAt hotelG_C cityG)
(hotelAt hotelH_A cityH)
(hotelAt hotelH_B cityH)
(hotelAt hotelH_C cityH)
(hotelAt hotelI_A cityI)
(hotelAt hotelI_B cityI)
(hotelAt hotelI_C cityI)
(hotelAt hotelJ_A cityJ)
(hotelAt hotelJ_B cityJ)
(hotelAt hotelJ_C cityJ)
(hotelAt hotelK_A cityK)
(hotelAt hotelK_B cityK)
(hotelAt hotelK_C cityK)
(hotelAt hotelL_A cityL)
(hotelAt hotelL_B cityL)
(hotelAt hotelL_C cityL)
(hotelAt hotelM_A cityM)
(hotelAt hotelM_B cityM)
(hotelAt hotelM_C cityM)
(hotelAt hotelN_A cityN)
(hotelAt hotelN_B cityN)
(hotelAt hotelN_C cityN)
(hotelAt hotelO_A cityO)
(hotelAt hotelO_B cityO)
(hotelAt hotelO_C cityO)
```

```
(flight Origin cityA)
(flight Origin cityB)
(flight Origin cityC)
(flight Origin cityD)
(flight Origin cityE)
(flight Origin cityF)
(flight Origin cityG)
(flight Origin cityH)
(flight Origin cityI)
(flight Origin cityJ)
(flight Origin cityK)
(flight Origin cityL)
(flight Origin cityM)
(flight Origin cityN)
(flight Origin cityO)
(flight cityA cityC)
(flight cityA cityF)
(flight cityA cityK)
(flight cityB cityC)
(flight cityB cityE)
(flight cityB cityN)
(flight cityC cityE)
(flight cityC cityF)
(flight cityC cityG)
(flight cityD cityG)
(flight cityD cityH)
(flight cityD cityO)
(flight cityE cityD)
(flight cityE cityI)
(flight cityE cityO)
(flight cityF cityL)
(flight cityF cityM)
(flight cityF cityN)
(flight cityG cityD)
(flight cityG cityF)
(flight cityG cityL)
(flight cityH cityD)
(flight cityH cityG)
(flight cityH cityN)
(flight cityI cityA)
(flight cityI cityG)
(flight cityI cityM)
(flight cityJ cityC)
(flight cityJ cityD)
(flight cityJ cityL)
(flight cityK cityB)
(flight cityK cityE)
```

```
(flight cityK cityI)
(flight cityL cityG)
(flight cityL cityI)
(flight cityL cityK)
(flight cityM cityB)
(flight cityM cityE)
(flight cityM cityH)
(flight cityN cityF)
(flight cityN cityH)
(flight cityN cityL)
(flight cityO cityH)
(flight cityO cityJ)
(flight cityO cityN)

(= (citiesVisited) 0)
(= (totalDays) 0)
(= (minDaysInCity) 2)
(= (maxDaysInCity) 4)

(= (stayDuration Origin) 0)
(= (stayDuration cityA) 0)
(= (stayDuration cityB) 0)
(= (stayDuration cityC) 0)
(= (stayDuration cityD) 0)
(= (stayDuration cityE) 0)
(= (stayDuration cityF) 0)
(= (stayDuration cityG) 0)
(= (stayDuration cityH) 0)
(= (stayDuration cityI) 0)
(= (stayDuration cityJ) 0)
(= (stayDuration cityK) 0)
(= (stayDuration cityL) 0)
(= (stayDuration cityM) 0)
(= (stayDuration cityN) 0)
(= (stayDuration cityO) 0)
(= (cityInterest Origin) 0)
(= (cityInterest cityA) 2)
(= (cityInterest cityB) 2)
(= (cityInterest cityC) 2)
(= (cityInterest cityD) 3)
(= (cityInterest cityE) 1)
(= (cityInterest cityF) 2)
(= (cityInterest cityG) 3)
(= (cityInterest cityH) 3)
(= (cityInterest cityI) 3)
(= (cityInterest cityJ) 3)
(= (cityInterest cityK) 2)
```

```
    (= (cityInterest cityL) 3)
    (= (cityInterest cityM) 2)
    (= (cityInterest cityN) 1)
    (= (cityInterest cityO) 2)
    (= (totalInterest) 0)

  )

  (:goal
    (and
      (>= (citiesVisited) 12)
      (>= (totalDays) 20)
    )
  )

  (:metric
    minimize (totalInterest)
  )

)
```

Salida:

```
ff: parsing domain file
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
 ... done.


metric established (normalized to minimize): ((1.00*[RF0](TOTALINTEREST)) - () + 0.00)

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is ((1.00*[RF0](TOTALINTEREST)) - () + 0.00)

advancing to distance:   12
                         11
                         10
                          9
                          8
                          7
                          6
                          5
```

```
                        4
                        3
                        2
                        1
                        0

ff: found legal plan as follows

step    0: FLYANDSTAY ORIGIN CITYB HOTELB_A
        1: FLYANDSTAY CITYB CITYC HOTELC_A
        2: FLYANDSTAY CITYC CITYF HOTELF_A
        3: FLYANDSTAY CITYF CITYM HOTELM_A
        4: INCREASESTAY CITYM
        5: INCREASESTAY CITYM
        6: FLYANDSTAY CITYM CITYH HOTELH_A
        7: FLYANDSTAY CITYH CITYD HOTELD_A
        8: FLYANDSTAY CITYD CITYO HOTELO_A
        9: INCREASESTAY CITYO
       10: INCREASESTAY CITYO
       11: FLYANDSTAY CITYO CITYN HOTELN_A
       12: FLYANDSTAY CITYN CITYL HOTELL_A
       13: FLYANDSTAY CITYL CITYK HOTELK_A
       14: INCREASESTAY CITYN
       15: INCREASESTAY CITYN
       16: FLYANDSTAY CITYK CITYI HOTELI_A
       17: FLYANDSTAY CITYI CITYA HOTELA_A


time spent:    0.00 seconds instantiating 196 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 47 facts and 195 actions
               0.00 seconds creating final representation with 46 relevant facts,
                                                          34 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.45 seconds searching, evaluating 22282 states, to a max depth of 0
               0.45 seconds total time
```

5.4.2 Prueba II

Entrada:

```
(define (problem autogen)
  (:domain worldsEndAndBeyond)
```

```
(:objects
  Origin cityA cityB cityC cityD cityE cityF cityG cityH cityI cityJ cityK
                                                              cityL - city
  hotelA_A hotelA_B hotelB_A hotelB_B hotelC_A hotelC_B hotelD_A
      hotelD_B hotelE_A hotelE_B hotelF_A hotelF_B hotelG_A hotelG_B
      hotelH_A hotelH_B hotelI_A hotelI_B hotelJ_A hotelJ_B hotelK_A
                                  hotelK_B hotelL_A hotelL_B - hotel
)

(:init
  (currentLocation Origin)

  (hotelAt hotelA_A cityA)
  (hotelAt hotelA_B cityA)
  (hotelAt hotelB_A cityB)
  (hotelAt hotelB_B cityB)
  (hotelAt hotelC_A cityC)
  (hotelAt hotelC_B cityC)
  (hotelAt hotelD_A cityD)
  (hotelAt hotelD_B cityD)
  (hotelAt hotelE_A cityE)
  (hotelAt hotelE_B cityE)
  (hotelAt hotelF_A cityF)
  (hotelAt hotelF_B cityF)
  (hotelAt hotelG_A cityG)
  (hotelAt hotelG_B cityG)
  (hotelAt hotelH_A cityH)
  (hotelAt hotelH_B cityH)
  (hotelAt hotelI_A cityI)
  (hotelAt hotelI_B cityI)
  (hotelAt hotelJ_A cityJ)
  (hotelAt hotelJ_B cityJ)
  (hotelAt hotelK_A cityK)
  (hotelAt hotelK_B cityK)
  (hotelAt hotelL_A cityL)
  (hotelAt hotelL_B cityL)

  (flight Origin cityA)
  (flight Origin cityB)
  (flight Origin cityC)
  (flight Origin cityD)
  (flight Origin cityE)
  (flight Origin cityF)
  (flight Origin cityG)
  (flight Origin cityH)
  (flight Origin cityI)
  (flight Origin cityJ)
```

```
(flight Origin cityK)
(flight Origin cityL)
(flight cityA cityB)
(flight cityA cityG)
(flight cityA cityH)
(flight cityA cityJ)
(flight cityB cityC)
(flight cityB cityH)
(flight cityB cityK)
(flight cityB cityL)
(flight cityC cityI)
(flight cityC cityJ)
(flight cityC cityK)
(flight cityC cityL)
(flight cityD cityA)
(flight cityD cityB)
(flight cityD cityK)
(flight cityD cityL)
(flight cityE cityA)
(flight cityE cityH)
(flight cityE cityI)
(flight cityE cityK)
(flight cityF cityA)
(flight cityF cityB)
(flight cityF cityH)
(flight cityF cityK)
(flight cityG cityA)
(flight cityG cityB)
(flight cityG cityF)
(flight cityG cityH)
(flight cityH cityE)
(flight cityH cityG)
(flight cityH cityI)
(flight cityH cityK)
(flight cityI cityC)
(flight cityI cityE)
(flight cityI cityF)
(flight cityI cityJ)
(flight cityJ cityC)
(flight cityJ cityE)
(flight cityJ cityF)
(flight cityJ cityK)
(flight cityK cityB)
(flight cityK cityG)
(flight cityK cityI)
(flight cityK cityJ)
(flight cityL cityB)
```

```
  (flight cityL cityF)
  (flight cityL cityG)
  (flight cityL cityJ)

  (= (citiesVisited) 0)
  (= (totalDays) 0)
  (= (minDaysInCity) 1)
  (= (maxDaysInCity) 2)

  (= (stayDuration Origin) 0)
  (= (stayDuration cityA) 0)
  (= (stayDuration cityB) 0)
  (= (stayDuration cityC) 0)
  (= (stayDuration cityD) 0)
  (= (stayDuration cityE) 0)
  (= (stayDuration cityF) 0)
  (= (stayDuration cityG) 0)
  (= (stayDuration cityH) 0)
  (= (stayDuration cityI) 0)
  (= (stayDuration cityJ) 0)
  (= (stayDuration cityK) 0)
  (= (stayDuration cityL) 0)
  (= (cityInterest Origin) 0)
  (= (cityInterest cityA) 1)
  (= (cityInterest cityB) 2)
  (= (cityInterest cityC) 2)
  (= (cityInterest cityD) 2)
  (= (cityInterest cityE) 1)
  (= (cityInterest cityF) 1)
  (= (cityInterest cityG) 2)
  (= (cityInterest cityH) 3)
  (= (cityInterest cityI) 1)
  (= (cityInterest cityJ) 3)
  (= (cityInterest cityK) 3)
  (= (cityInterest cityL) 2)
  (= (totalInterest) 0)

)

(:goal
  (and
    (>= (citiesVisited) 6)
    (>= (totalDays) 20)
  )
)

(:metric
```

```
    minimize (totalInterest)
  )

)
```

Salida:

```
ff: parsing domain file
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
 ... done.


metric established (normalized to minimize): ((1.00*[RF0](TOTALINTEREST)) - () + 0.00)

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is ((1.00*[RF0](TOTALINTEREST)) - () + 0.00)

advancing to distance:   20
                         19
                         18
                         17
                         16
                         15
                         14
                         13
                         12
                         11
                         10
                          9
                          8
                          7
                          6
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows
```

```
step    0: FLYANDSTAY ORIGIN CITYH HOTELH_A
        1: FLYANDSTAY CITYH CITYE HOTELE_A
        2: FLYANDSTAY CITYE CITYK HOTELK_A
        3: FLYANDSTAY CITYK CITYJ HOTELJ_A
        4: FLYANDSTAY CITYJ CITYC HOTELC_A
        5: FLYANDSTAY CITYC CITYI HOTELI_A
        6: INCREASESTAY CITYK
        7: FLYANDSTAY CITYI CITYF HOTELF_A
        8: FLYANDSTAY CITYF CITYA HOTELA_A
        9: FLYANDSTAY CITYA CITYG HOTELG_A
       10: INCREASESTAY CITYA
       11: INCREASESTAY CITYC
       12: INCREASESTAY CITYE
       13: INCREASESTAY CITYF
       14: INCREASESTAY CITYG
       15: INCREASESTAY CITYH
       16: INCREASESTAY CITYI
       17: INCREASESTAY CITYJ
       18: FLYANDSTAY CITYG CITYB HOTELB_A
       19: INCREASESTAY CITYB


time spent:    0.00 seconds instantiating 133 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 38 facts and 132 actions
               0.00 seconds creating final representation with 37 relevant facts,
                                                              28 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 229 states, to a max depth of 0
               0.00 seconds total time
```

5.5 Pruebas de la extensión 3

En las pruebas de esta extensión, partiendo de los tamaños anteriores, hemos tenido que reducir el número de elementos del problema porque sino nos pasábamos del límite de líneas y nos daba problemas la ejecución. Dicho esto, para el resto hemos procedido de la misma forma que antes, y de nuevo observamos unos resultados que cumplen las restricciones y obtenidos rápidamente. Esta vez, recordamos que se debe introducir los costes. Los precios son generados aleatoriamente por el script, con una cierta desviación y dentro de un rango.

5.5.1 Prueba I

Entrada:

```
(define (problem autogen)
  (:domain worldsEndAndBeyond)

  (:objects
    Origin cityA cityB cityC cityD cityE cityF cityG cityH cityI - city
    hotelA_A hotelA_B hotelA_C hotelB_A hotelB_B hotelB_C hotelC_A
        hotelC_B hotelC_C hotelD_A hotelD_B hotelD_C hotelE_A hotelE_B
        hotelE_C hotelF_A hotelF_B hotelF_C hotelG_A hotelG_B hotelG_C
        hotelH_A hotelH_B hotelH_C hotelI_A hotelI_B hotelI_C - hotel
  )

  (:init
    (currentLocation Origin)

    (hotelAt hotelA_A cityA)
    (hotelAt hotelA_B cityA)
    (hotelAt hotelA_C cityA)
    (hotelAt hotelB_A cityB)
    (hotelAt hotelB_B cityB)
    (hotelAt hotelB_C cityB)
    (hotelAt hotelC_A cityC)
    (hotelAt hotelC_B cityC)
    (hotelAt hotelC_C cityC)
    (hotelAt hotelD_A cityD)
    (hotelAt hotelD_B cityD)
    (hotelAt hotelD_C cityD)
    (hotelAt hotelE_A cityE)
    (hotelAt hotelE_B cityE)
    (hotelAt hotelE_C cityE)
    (hotelAt hotelF_A cityF)
    (hotelAt hotelF_B cityF)
    (hotelAt hotelF_C cityF)
    (hotelAt hotelG_A cityG)
    (hotelAt hotelG_B cityG)
    (hotelAt hotelG_C cityG)
    (hotelAt hotelH_A cityH)
    (hotelAt hotelH_B cityH)
    (hotelAt hotelH_C cityH)
    (hotelAt hotelI_A cityI)
    (hotelAt hotelI_B cityI)
    (hotelAt hotelI_C cityI)

    (flight Origin cityA)
    (flight Origin cityB)
    (flight Origin cityC)
    (flight Origin cityD)
    (flight Origin cityE)
```

```
(flight Origin cityF)
(flight Origin cityG)
(flight Origin cityH)
(flight Origin cityI)
(flight cityA cityE)
(flight cityA cityF)
(flight cityA cityI)
(flight cityB cityD)
(flight cityB cityE)
(flight cityB cityF)
(flight cityC cityE)
(flight cityC cityF)
(flight cityC cityH)
(flight cityD cityC)
(flight cityD cityG)
(flight cityD cityH)
(flight cityE cityB)
(flight cityE cityD)
(flight cityE cityG)
(flight cityF cityA)
(flight cityF cityB)
(flight cityF cityH)
(flight cityG cityA)
(flight cityG cityD)
(flight cityG cityI)
(flight cityH cityA)
(flight cityH cityD)
(flight cityH cityF)
(flight cityI cityF)
(flight cityI cityG)
(flight cityI cityH)

(= (citiesVisited) 0)
(= (totalDays) 0)
(= (minDaysInCity) 2)
(= (maxDaysInCity) 3)

(= (stayDuration Origin) 0)
(= (stayDuration cityA) 0)
(= (stayDuration cityB) 0)
(= (stayDuration cityC) 0)
(= (stayDuration cityD) 0)
(= (stayDuration cityE) 0)
(= (stayDuration cityF) 0)
(= (stayDuration cityG) 0)
(= (stayDuration cityH) 0)
(= (stayDuration cityI) 0)
```

```
(= (totalcost) 0)
(= (hotelcost hotelA_A) 207)
(= (hotelcost hotelA_B) 161)
(= (hotelcost hotelA_C) 158)
(= (hotelcost hotelB_A) 134)
(= (hotelcost hotelB_B) 109)
(= (hotelcost hotelB_C) 161)
(= (hotelcost hotelC_A) 161)
(= (hotelcost hotelC_B) 129)
(= (hotelcost hotelC_C) 191)
(= (hotelcost hotelD_A) 138)
(= (hotelcost hotelD_B) 83)
(= (hotelcost hotelD_C) 229)
(= (hotelcost hotelE_A) 110)
(= (hotelcost hotelE_B) 87)
(= (hotelcost hotelE_C) 156)
(= (hotelcost hotelF_A) 227)
(= (hotelcost hotelF_B) 119)
(= (hotelcost hotelF_C) 88)
(= (hotelcost hotelG_A) 170)
(= (hotelcost hotelG_B) 54)
(= (hotelcost hotelG_C) 122)
(= (hotelcost hotelH_A) 194)
(= (hotelcost hotelH_B) 113)
(= (hotelcost hotelH_C) 232)
(= (hotelcost hotelI_A) 181)
(= (hotelcost hotelI_B) 228)
(= (hotelcost hotelI_C) 202)

(= (flightcost Origin cityA) 0)
(= (flightcost Origin cityB) 0)
(= (flightcost Origin cityC) 0)
(= (flightcost Origin cityD) 0)
(= (flightcost Origin cityE) 0)
(= (flightcost Origin cityF) 0)
(= (flightcost Origin cityG) 0)
(= (flightcost Origin cityH) 0)
(= (flightcost Origin cityI) 0)
(= (flightcost cityA cityE) 184)
(= (flightcost cityA cityF) 153)
(= (flightcost cityA cityI) 147)
(= (flightcost cityB cityD) 151)
(= (flightcost cityB cityE) 110)
(= (flightcost cityB cityF) 110)
(= (flightcost cityC cityE) 112)
(= (flightcost cityC cityF) 147)
(= (flightcost cityC cityH) 169)
```

```
    (= (flightcost cityD cityC) 123)
    (= (flightcost cityD cityG) 158)
    (= (flightcost cityD cityH) 101)
    (= (flightcost cityE cityB) 164)
    (= (flightcost cityE cityD) 198)
    (= (flightcost cityE cityG) 186)
    (= (flightcost cityF cityA) 195)
    (= (flightcost cityF cityB) 110)
    (= (flightcost cityF cityH) 123)
    (= (flightcost cityG cityA) 102)
    (= (flightcost cityG cityD) 139)
    (= (flightcost cityG cityI) 145)
    (= (flightcost cityH cityA) 192)
    (= (flightcost cityH cityD) 159)
    (= (flightcost cityH cityF) 101)
    (= (flightcost cityI cityF) 165)
    (= (flightcost cityI cityG) 156)
    (= (flightcost cityI cityH) 116)

  )

  (:goal
    (and
      (>= (citiesVisited) 7)
      (>= (totalDays) 10)
      (>= (totalcost) 2000)
      (<= (totalcost) 3000)
    )
  )

  (:metric
    minimize (totalcost)
  )

)
```

Salida:

```
ff: parsing domain file
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
 ... done.
```

```
metric established (normalized to minimize): ((1.00*[RF0](TOTALCOST)) - () + 0.00)

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is ((1.00*[RF0](TOTALCOST)) - () + 0.00)

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

ff: found legal plan as follows

step    0: FLYANDSTAY ORIGIN CITYB HOTELB_B
        1: FLYANDSTAY CITYB CITYE HOTELE_B
        2: FLYANDSTAY CITYE CITYD HOTELD_A
        3: FLYANDSTAY CITYD CITYG HOTELG_A
        4: FLYANDSTAY CITYG CITYA HOTELA_C
        5: FLYANDSTAY CITYA CITYF HOTELF_C
        6: FLYANDSTAY CITYF CITYH HOTELH_B


time spent:    0.00 seconds instantiating 135 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 56 facts and 135 actions
               0.00 seconds creating final representation with 55 relevant facts,
                                                          23 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 88 states, to a max depth of 0
               0.00 seconds total time
```

5.5.2 Prueba II

Entrada:

```
(define (problem autogen)
  (:domain worldsEndAndBeyond)

  (:objects
```

```
  Origin cityA cityB cityC cityD cityE cityF cityG cityH - city
  hotelA_A hotelA_B hotelB_A hotelB_B hotelC_A hotelC_B hotelD_A
      hotelD_B hotelE_A hotelE_B hotelF_A hotelF_B hotelG_A hotelG_B
                                              hotelH_A hotelH_B - hotel
)

(:init
  (currentLocation Origin)

  (hotelAt hotelA_A cityA)
  (hotelAt hotelA_B cityA)
  (hotelAt hotelB_A cityB)
  (hotelAt hotelB_B cityB)
  (hotelAt hotelC_A cityC)
  (hotelAt hotelC_B cityC)
  (hotelAt hotelD_A cityD)
  (hotelAt hotelD_B cityD)
  (hotelAt hotelE_A cityE)
  (hotelAt hotelE_B cityE)
  (hotelAt hotelF_A cityF)
  (hotelAt hotelF_B cityF)
  (hotelAt hotelG_A cityG)
  (hotelAt hotelG_B cityG)
  (hotelAt hotelH_A cityH)
  (hotelAt hotelH_B cityH)

  (flight Origin cityA)
  (flight Origin cityB)
  (flight Origin cityC)
  (flight Origin cityD)
  (flight Origin cityE)
  (flight Origin cityF)
  (flight Origin cityG)
  (flight Origin cityH)
  (flight cityA cityB)
  (flight cityA cityC)
  (flight cityA cityE)
  (flight cityA cityG)
  (flight cityB cityA)
  (flight cityB cityD)
  (flight cityB cityF)
  (flight cityB cityG)
  (flight cityC cityB)
  (flight cityC cityE)
  (flight cityC cityG)
  (flight cityC cityH)
  (flight cityD cityE)
```

```
(flight cityD cityF)
(flight cityD cityG)
(flight cityD cityH)
(flight cityE cityD)
(flight cityE cityF)
(flight cityE cityG)
(flight cityE cityH)
(flight cityF cityC)
(flight cityF cityD)
(flight cityF cityE)
(flight cityF cityG)
(flight cityG cityD)
(flight cityG cityE)
(flight cityG cityF)
(flight cityG cityH)
(flight cityH cityB)
(flight cityH cityC)
(flight cityH cityD)
(flight cityH cityE)

(= (citiesVisited) 0)
(= (totalDays) 0)
(= (minDaysInCity) 1)
(= (maxDaysInCity) 2)

(= (stayDuration Origin) 0)
(= (stayDuration cityA) 0)
(= (stayDuration cityB) 0)
(= (stayDuration cityC) 0)
(= (stayDuration cityD) 0)
(= (stayDuration cityE) 0)
(= (stayDuration cityF) 0)
(= (stayDuration cityG) 0)
(= (stayDuration cityH) 0)
(= (totalcost) 0)
(= (hotelcost hotelA_A) 186)
(= (hotelcost hotelA_B) 93)
(= (hotelcost hotelB_A) 164)
(= (hotelcost hotelB_B) 217)
(= (hotelcost hotelC_A) 104)
(= (hotelcost hotelC_B) 143)
(= (hotelcost hotelD_A) 157)
(= (hotelcost hotelD_B) 245)
(= (hotelcost hotelE_A) 106)
(= (hotelcost hotelE_B) 228)
(= (hotelcost hotelF_A) 169)
(= (hotelcost hotelF_B) 193)
```

```
  (= (hotelcost hotelG_A) 210)
  (= (hotelcost hotelG_B) 125)
  (= (hotelcost hotelH_A) 102)
  (= (hotelcost hotelH_B) 208)

  (= (flightcost Origin cityA) 0)
  (= (flightcost Origin cityB) 0)
  (= (flightcost Origin cityC) 0)
  (= (flightcost Origin cityD) 0)
  (= (flightcost Origin cityE) 0)
  (= (flightcost Origin cityF) 0)
  (= (flightcost Origin cityG) 0)
  (= (flightcost Origin cityH) 0)
  (= (flightcost cityA cityB) 126)
  (= (flightcost cityA cityC) 196)
  (= (flightcost cityA cityE) 143)
  (= (flightcost cityA cityG) 121)
  (= (flightcost cityB cityA) 167)
  (= (flightcost cityB cityD) 105)
  (= (flightcost cityB cityF) 166)
  (= (flightcost cityB cityG) 114)
  (= (flightcost cityC cityB) 145)
  (= (flightcost cityC cityE) 178)
  (= (flightcost cityC cityG) 165)
  (= (flightcost cityC cityH) 103)
  (= (flightcost cityD cityE) 137)
  (= (flightcost cityD cityF) 159)
  (= (flightcost cityD cityG) 103)
  (= (flightcost cityD cityH) 173)
  (= (flightcost cityE cityD) 154)
  (= (flightcost cityE cityF) 169)
  (= (flightcost cityE cityG) 140)
  (= (flightcost cityE cityH) 108)
  (= (flightcost cityF cityC) 162)
  (= (flightcost cityF cityD) 199)
  (= (flightcost cityF cityE) 103)
  (= (flightcost cityF cityG) 170)
  (= (flightcost cityG cityD) 177)
  (= (flightcost cityG cityE) 175)
  (= (flightcost cityG cityF) 165)
  (= (flightcost cityG cityH) 189)
  (= (flightcost cityH cityB) 102)
  (= (flightcost cityH cityC) 117)
  (= (flightcost cityH cityD) 199)
  (= (flightcost cityH cityE) 180)

)
```

```
  (:goal
    (and
      (>= (citiesVisited) 4)
      (>= (totalDays) 10)
      (>= (totalcost) 1500)
      (<= (totalcost) 2000)
    )
  )

  (:metric
    minimize (totalcost)
  )

)
```

Salida:

```
ff: parsing domain file
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
 ... done.


metric established (normalized to minimize): ((1.00*[RF0](TOTALCOST)) - () + 0.00)

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is ((1.00*[RF0](TOTALCOST)) - () + 0.00)

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

ff: found legal plan as follows
```

```
step    0: FLYANDSTAY ORIGIN CITYA HOTELA_B
        1: FLYANDSTAY CITYA CITYB HOTELB_A
        2: INCREASESTAY CITYB HOTELB_A
        3: INCREASESTAY CITYA HOTELA_B
        4: FLYANDSTAY CITYB CITYD HOTELD_A
        5: FLYANDSTAY CITYD CITYF HOTELF_A
        6: FLYANDSTAY CITYF CITYE HOTELE_A
        7: FLYANDSTAY CITYE CITYH HOTELH_A
        8: INCREASESTAY CITYH HOTELH_A
        9: INCREASESTAY CITYE HOTELE_A


time spent:    0.00 seconds instantiating 96 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 42 facts and 96 actions
               0.00 seconds creating final representation with 41 relevant facts,
                                                              21 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 82 states, to a max depth of 0
               0.00 seconds total time
```

5.6 Pruebas de la extensión 4

Para las pruebas de esta versión, cabe destacar que hemos tenido que introducir la ponderación (que se indica en los ficheros de problemas). El programa generador de problemas calcula automáticamente el peso de interés siguiendo la fórmula que hemos indicado en el modelado de problemas. Con otras pruebas anteriores a las presentadas (las definitivas), vimos que dábamos un excesivo peso al interés y entonces las soluciones daban demasiada poca relevancia a los precios.

Ajustando la ponderación, pues, podríamos dar más importancia a uno u otro criterio, pero hemos intentado equiparar la relevancia de ambos de forma algo arbitraria (porque al fin y al cabo son unidades distintas), pero razonable a nuestro entender.

5.6.1 Prueba I

Entrada:

```
define (problem autogen)
  (:domain worldsEndAndBeyond)

  (:objects
    Origin cityA cityB cityC cityD cityE cityF cityG cityH cityI - city
```

```
  hotelA_A hotelA_B hotelA_C hotelB_A hotelB_B hotelB_C hotelC_A
    hotelC_B hotelC_C hotelD_A hotelD_B hotelD_C hotelE_A hotelE_B
    hotelE_C hotelF_A hotelF_B hotelF_C hotelG_A hotelG_B hotelG_C
     hotelH_A hotelH_B hotelH_C hotelI_A hotelI_B hotelI_C - hotel
)

(:init
  (currentLocation Origin)

  (hotelAt hotelA_A cityA)
  (hotelAt hotelA_B cityA)
  (hotelAt hotelA_C cityA)
  (hotelAt hotelB_A cityB)
  (hotelAt hotelB_B cityB)
  (hotelAt hotelB_C cityB)
  (hotelAt hotelC_A cityC)
  (hotelAt hotelC_B cityC)
  (hotelAt hotelC_C cityC)
  (hotelAt hotelD_A cityD)
  (hotelAt hotelD_B cityD)
  (hotelAt hotelD_C cityD)
  (hotelAt hotelE_A cityE)
  (hotelAt hotelE_B cityE)
  (hotelAt hotelE_C cityE)
  (hotelAt hotelF_A cityF)
  (hotelAt hotelF_B cityF)
  (hotelAt hotelF_C cityF)
  (hotelAt hotelG_A cityG)
  (hotelAt hotelG_B cityG)
  (hotelAt hotelG_C cityG)
  (hotelAt hotelH_A cityH)
  (hotelAt hotelH_B cityH)
  (hotelAt hotelH_C cityH)
  (hotelAt hotelI_A cityI)
  (hotelAt hotelI_B cityI)
  (hotelAt hotelI_C cityI)

  (flight Origin cityA)
  (flight Origin cityB)
  (flight Origin cityC)
  (flight Origin cityD)
  (flight Origin cityE)
  (flight Origin cityF)
  (flight Origin cityG)
  (flight Origin cityH)
  (flight Origin cityI)
  (flight cityA cityC)
```

```
(flight cityA cityD)
(flight cityA cityH)
(flight cityB cityC)
(flight cityB cityD)
(flight cityB cityH)
(flight cityC cityA)
(flight cityC cityD)
(flight cityC cityF)
(flight cityD cityA)
(flight cityD cityB)
(flight cityD cityG)
(flight cityE cityC)
(flight cityE cityG)
(flight cityE cityH)
(flight cityF cityB)
(flight cityF cityD)
(flight cityF cityI)
(flight cityG cityB)
(flight cityG cityC)
(flight cityG cityE)
(flight cityH cityB)
(flight cityH cityE)
(flight cityH cityG)
(flight cityI cityE)
(flight cityI cityG)
(flight cityI cityH)

(= (citiesVisited) 0)
(= (totalDays) 0)
(= (minDaysInCity) 2)
(= (maxDaysInCity) 3)

(= (stayDuration Origin) 0)
(= (stayDuration cityA) 0)
(= (stayDuration cityB) 0)
(= (stayDuration cityC) 0)
(= (stayDuration cityD) 0)
(= (stayDuration cityE) 0)
(= (stayDuration cityF) 0)
(= (stayDuration cityG) 0)
(= (stayDuration cityH) 0)
(= (stayDuration cityI) 0)
(= (cityInterest Origin) 0)
(= (cityInterest cityA) 2)
(= (cityInterest cityB) 3)
(= (cityInterest cityC) 1)
(= (cityInterest cityD) 2)
```

```
(= (cityInterest cityE) 3)
(= (cityInterest cityF) 1)
(= (cityInterest cityG) 2)
(= (cityInterest cityH) 1)
(= (cityInterest cityI) 2)
(= (totalInterest) 0)
(= (totalcost) 0)
(= (hotelcost hotelA_A) 145)
(= (hotelcost hotelA_B) 203)
(= (hotelcost hotelA_C) 159)
(= (hotelcost hotelB_A) 219)
(= (hotelcost hotelB_B) 247)
(= (hotelcost hotelB_C) 85)
(= (hotelcost hotelC_A) 60)
(= (hotelcost hotelC_B) 202)
(= (hotelcost hotelC_C) 180)
(= (hotelcost hotelD_A) 56)
(= (hotelcost hotelD_B) 216)
(= (hotelcost hotelD_C) 168)
(= (hotelcost hotelE_A) 153)
(= (hotelcost hotelE_B) 193)
(= (hotelcost hotelE_C) 124)
(= (hotelcost hotelF_A) 138)
(= (hotelcost hotelF_B) 180)
(= (hotelcost hotelF_C) 144)
(= (hotelcost hotelG_A) 163)
(= (hotelcost hotelG_B) 236)
(= (hotelcost hotelG_C) 124)
(= (hotelcost hotelH_A) 78)
(= (hotelcost hotelH_B) 212)
(= (hotelcost hotelH_C) 148)
(= (hotelcost hotelI_A) 208)
(= (hotelcost hotelI_B) 134)
(= (hotelcost hotelI_C) 105)

(= (flightcost Origin cityA) 0)
(= (flightcost Origin cityB) 0)
(= (flightcost Origin cityC) 0)
(= (flightcost Origin cityD) 0)
(= (flightcost Origin cityE) 0)
(= (flightcost Origin cityF) 0)
(= (flightcost Origin cityG) 0)
(= (flightcost Origin cityH) 0)
(= (flightcost Origin cityI) 0)
(= (flightcost cityA cityC) 105)
(= (flightcost cityA cityD) 164)
(= (flightcost cityA cityH) 129)
```

```
    (= (flightcost cityB cityC) 138)
    (= (flightcost cityB cityD) 111)
    (= (flightcost cityB cityH) 182)
    (= (flightcost cityC cityA) 147)
    (= (flightcost cityC cityD) 132)
    (= (flightcost cityC cityF) 180)
    (= (flightcost cityD cityA) 135)
    (= (flightcost cityD cityB) 143)
    (= (flightcost cityD cityG) 184)
    (= (flightcost cityE cityC) 165)
    (= (flightcost cityE cityG) 101)
    (= (flightcost cityE cityH) 150)
    (= (flightcost cityF cityB) 135)
    (= (flightcost cityF cityD) 104)
    (= (flightcost cityF cityI) 145)
    (= (flightcost cityG cityB) 110)
    (= (flightcost cityG cityC) 144)
    (= (flightcost cityG cityE) 176)
    (= (flightcost cityH cityB) 156)
    (= (flightcost cityH cityE) 158)
    (= (flightcost cityH cityG) 114)
    (= (flightcost cityI cityE) 183)
    (= (flightcost cityI cityG) 186)
    (= (flightcost cityI cityH) 176)

  )

  (:goal
    (and
      (>= (citiesVisited) 7)
      (>= (totalDays) 10)
      (>= (totalcost) 2000)
      (<= (totalcost) 3000)
    )
  )

  (:metric
    minimize
      (+ (* 255 (totalInterest)) (totalcost))
  )

)
```

Salida:

```
ff: parsing domain file
```

```
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
 ... done.


metric established (normalized to minimize):
((255.00*[RF1](TOTALINTEREST)1.00*[RF0](TOTALCOST)) - () + 0.00)

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is ((255.00*[RF1](TOTALINTEREST)1.00*[RF0](TOTALCOST)) - () + 0.00)

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

ff: found legal plan as follows

step    0: FLYANDSTAY ORIGIN CITYA HOTELA_B
        1: FLYANDSTAY CITYA CITYC HOTELC_A
        2: FLYANDSTAY CITYC CITYF HOTELF_A
        3: FLYANDSTAY CITYF CITYB HOTELB_C
        4: FLYANDSTAY CITYB CITYD HOTELD_A
        5: FLYANDSTAY CITYD CITYG HOTELG_C
        6: FLYANDSTAY CITYG CITYE HOTELE_C


time spent:    0.00 seconds instantiating 135 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 56 facts and 135 actions
               0.00 seconds creating final representation with 55 relevant facts,
                                                          24 relevant fluents
               0.00 seconds computing LNF
               0.01 seconds building connectivity graph
               0.00 seconds searching, evaluating 181 states, to a max depth of 0
               0.01 seconds total time
```

5.6.2 Prueba II

Entrada:

```
(define (problem autogen)
  (:domain worldsEndAndBeyond)

  (:objects
    Origin cityA cityB cityC cityD cityE cityF cityG cityH - city
    hotelA_A hotelA_B hotelB_A hotelB_B hotelC_A hotelC_B hotelD_A
        hotelD_B hotelE_A hotelE_B hotelF_A hotelF_B hotelG_A hotelG_B
                                                hotelH_A hotelH_B - hotel
  )

  (:init
    (currentLocation Origin)

    (hotelAt hotelA_A cityA)
    (hotelAt hotelA_B cityA)
    (hotelAt hotelB_A cityB)
    (hotelAt hotelB_B cityB)
    (hotelAt hotelC_A cityC)
    (hotelAt hotelC_B cityC)
    (hotelAt hotelD_A cityD)
    (hotelAt hotelD_B cityD)
    (hotelAt hotelE_A cityE)
    (hotelAt hotelE_B cityE)
    (hotelAt hotelF_A cityF)
    (hotelAt hotelF_B cityF)
    (hotelAt hotelG_A cityG)
    (hotelAt hotelG_B cityG)
    (hotelAt hotelH_A cityH)
    (hotelAt hotelH_B cityH)

    (flight Origin cityA)
    (flight Origin cityB)
    (flight Origin cityC)
    (flight Origin cityD)
    (flight Origin cityE)
    (flight Origin cityF)
    (flight Origin cityG)
    (flight Origin cityH)
    (flight cityA cityB)
    (flight cityA cityD)
    (flight cityA cityE)
    (flight cityA cityF)
    (flight cityB cityA)
```

```
(flight cityB cityD)
(flight cityB cityG)
(flight cityB cityH)
(flight cityC cityA)
(flight cityC cityB)
(flight cityC cityF)
(flight cityC cityG)
(flight cityD cityB)
(flight cityD cityC)
(flight cityD cityG)
(flight cityD cityH)
(flight cityE cityA)
(flight cityE cityC)
(flight cityE cityF)
(flight cityE cityG)
(flight cityF cityA)
(flight cityF cityC)
(flight cityF cityE)
(flight cityF cityH)
(flight cityG cityA)
(flight cityG cityC)
(flight cityG cityE)
(flight cityG cityF)
(flight cityH cityA)
(flight cityH cityB)
(flight cityH cityC)
(flight cityH cityG)

(= (citiesVisited) 0)
(= (totalDays) 0)
(= (minDaysInCity) 1)
(= (maxDaysInCity) 2)

(= (stayDuration Origin) 0)
(= (stayDuration cityA) 0)
(= (stayDuration cityB) 0)
(= (stayDuration cityC) 0)
(= (stayDuration cityD) 0)
(= (stayDuration cityE) 0)
(= (stayDuration cityF) 0)
(= (stayDuration cityG) 0)
(= (stayDuration cityH) 0)
(= (cityInterest Origin) 0)
(= (cityInterest cityA) 3)
(= (cityInterest cityB) 1)
(= (cityInterest cityC) 1)
(= (cityInterest cityD) 1)
```

```
(= (cityInterest cityE) 2)
(= (cityInterest cityF) 2)
(= (cityInterest cityG) 3)
(= (cityInterest cityH) 3)
(= (totalInterest) 0)
(= (totalcost) 0)
(= (hotelcost hotelA_A) 233)
(= (hotelcost hotelA_B) 117)
(= (hotelcost hotelB_A) 147)
(= (hotelcost hotelB_B) 231)
(= (hotelcost hotelC_A) 177)
(= (hotelcost hotelC_B) 50)
(= (hotelcost hotelD_A) 177)
(= (hotelcost hotelD_B) 90)
(= (hotelcost hotelE_A) 176)
(= (hotelcost hotelE_B) 239)
(= (hotelcost hotelF_A) 226)
(= (hotelcost hotelF_B) 100)
(= (hotelcost hotelG_A) 159)
(= (hotelcost hotelG_B) 202)
(= (hotelcost hotelH_A) 240)
(= (hotelcost hotelH_B) 201)

(= (flightcost Origin cityA) 0)
(= (flightcost Origin cityB) 0)
(= (flightcost Origin cityC) 0)
(= (flightcost Origin cityD) 0)
(= (flightcost Origin cityE) 0)
(= (flightcost Origin cityF) 0)
(= (flightcost Origin cityG) 0)
(= (flightcost Origin cityH) 0)
(= (flightcost cityA cityB) 189)
(= (flightcost cityA cityD) 149)
(= (flightcost cityA cityE) 185)
(= (flightcost cityA cityF) 164)
(= (flightcost cityB cityA) 159)
(= (flightcost cityB cityD) 142)
(= (flightcost cityB cityG) 104)
(= (flightcost cityB cityH) 152)
(= (flightcost cityC cityA) 169)
(= (flightcost cityC cityB) 152)
(= (flightcost cityC cityF) 184)
(= (flightcost cityC cityG) 172)
(= (flightcost cityD cityB) 130)
(= (flightcost cityD cityC) 189)
(= (flightcost cityD cityG) 139)
(= (flightcost cityD cityH) 113)
```

```
    (= (flightcost cityE cityA) 156)
    (= (flightcost cityE cityC) 136)
    (= (flightcost cityE cityF) 146)
    (= (flightcost cityE cityG) 136)
    (= (flightcost cityF cityA) 188)
    (= (flightcost cityF cityC) 174)
    (= (flightcost cityF cityE) 176)
    (= (flightcost cityF cityH) 114)
    (= (flightcost cityG cityA) 115)
    (= (flightcost cityG cityC) 152)
    (= (flightcost cityG cityE) 164)
    (= (flightcost cityG cityF) 124)
    (= (flightcost cityH cityA) 157)
    (= (flightcost cityH cityB) 155)
    (= (flightcost cityH cityC) 128)
    (= (flightcost cityH cityG) 146)

  )

  (:goal
    (and
      (>= (citiesVisited) 4)
      (>= (totalDays) 10)
      (>= (totalcost) 1500)
      (<= (totalcost) 2000)
    )
  )

  (:metric
    minimize
      (+ (* 350 (totalInterest)) (totalcost))
  )

)
```

Salida:

```
ff: parsing domain file
domain 'WORLDSENDANDBEYOND' defined
 ... done.
ff: parsing problem file
problem 'AUTOGEN' defined
 ... done.


metric established (normalized to minimize):
```

```
((350.00*[RF1](TOTALINTEREST)1.00*[RF0](TOTALCOST)) - () + 0.00)

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is ((350.00*[RF1](TOTALINTEREST)1.00*[RF0](TOTALCOST)) - () + 0.00)

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

ff: found legal plan as follows

step    0: FLYANDSTAY ORIGIN CITYA HOTELA_B
        1: FLYANDSTAY CITYA CITYE HOTELE_A
        2: FLYANDSTAY CITYE CITYG HOTELG_A
        3: INCREASESTAY CITYE HOTELE_A
        4: INCREASESTAY CITYG HOTELG_A
        5: INCREASESTAY CITYA HOTELA_B
        6: FLYANDSTAY CITYG CITYC HOTELC_B
        7: FLYANDSTAY CITYC CITYB HOTELB_A
        8: INCREASESTAY CITYC HOTELC_B
        9: INCREASESTAY CITYB HOTELB_A


time spent:    0.00 seconds instantiating 96 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 42 facts and 96 actions
               0.00 seconds creating final representation with 41 relevant facts,
                                                              22 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.01 seconds searching, evaluating 72 states, to a max depth of 0
               0.01 seconds total time
```

5.7 Estudio de la evolución del tiempo de ejecución con respecto al tamaño del problema

Utilizando el programa generador de problemas para la extensión 3, hemos creado automáticamente problemas con un tamaño distinto. Concretamente, hemos ido aumentando el número de ciudades de 5 en cinco, partiendo de 5 y acabando en 25, intentando mantener el resto constante. El problema con el que nos hemos encontrado es el límite de 200 líneas, que ha hecho que tuviéramos que reducir el número de algunos de los elementos de los problemas con más ciudades. Además, en el caso de 25 ciudades, directamente el programa no terminaba la ejecución.

Todos los datos están adjuntos en la entrega, pero en este documento mostraremos los resultados mediante una tabla:

ciudades	5	10	15	20
tiempo (s)	0.00	0.18	0.00	2.69

Observamos que el tiempo de mantiene relativamente bajo (alrededor de 0.00-0.20 segundos), con la excepción de 20 ciudades caso en el que no encuentra solución y se le termina el espacio de búsqueda. Remarcamos que en el caso de 20 ciudades ocurre esto debido a que se tuvo que bajar el número de vuelos por ciudad de 2 a 1 debido al límite de 200 líneas mencionado.

6 Conclusiones

Una vez hemos modelado y desarrollado el dominio y los problemas, probado varios casos no triviales y estudiado la evolución del tiempo de generación de una solución con respecto al tamaño de cada problema, todo ello recopilado y documentado en esta memoria, llegamos a las siguientes conclusiones:

- Hemos sido capaces de resolver satisfactoriamente un problema de síntesis, el de las recomendaciones de viajes para la agencia de viajes Al Fin Del Mundo Y Más Allá, con un planificador.
- Respecto a las pruebas, consideramos que los resultados validan el correcto funcionamiento del programa en casos con una cierta complejidad. Sin embargo, no han sido viables las pruebas con mucha más complejidad de la incluida en esta documentación por el límite de líneas del programa.
- Como nota, insistimos, nos ha llamado la atención el límite de 200 líneas y consideramos que la mayoría de problemas del mundo real requerirían que se arreglara este error.
- En cuanto al estudio de la evolución del tiempo de resolución con respecto al tamaño del problema, observamos que al aumentar las ciudades no parece aumentar significativemente el tiempo de ejecución dentro de los parámetros que hemos podido probar dada la restricción mencionada en el punto anterior.
- Con esta práctica, también nos hemos dado cuenta de que, efectivamente, las optimizaciones de Fast Forward son greedy, y en varias ocasiones puede dar resultados relativamente lejos del óptimo global.
- Finalmente, en cuanto a la ponderación usada para la extensión 4 cabe decir que hemos considerado distintos pesos pero hemos optado por el que nos parecía que más equilibraba la importancia del interés y la del coste. Obviamente, ajustando esta ponderación podríamos hacer que las soluciones se generaran dando más relevancia a uno o a otro criterio. Podría ser al gusto del usuario o según la opinión de la agencia de viajes, más conocedora del dominio o de sus intereses económicos.