UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

TRABAJO FINAL DE MÁSTER

PLANIFICACIÓN INTELIGENTE DE RUTAS TURÍSTICAS EN LA CIUDAD DE VALENCIA

MUIARFID

Presentado por:
Jesús Ibáñez Ruiz

Dirigido por:
Dra. Laura Sebastiá Tarín
Dra. Eva Onaindia de la Rivaherrera

Valencia, 2015

Dedicatoria
A mi familia.

Resumen

Este proyecto describe Valencia Tourism Planner (VTP), una aplicación que utiliza un sistema de recomendación y un planificador automatizado para generar rutas automáticas y personalizadas en Valencia. La personalización se produce en dos etapas: primero, el sistema de recomendación obtiene una lista de actividades de acuerdo con el perfil de usuario y, a continuación, el usuario puede especificar algunas preferencias con respecto a la configuración de la agenda. Este documento describe en detalle el diseño de la aplicación y cómo el problema de planificación se construye en base las preferencias del usuario. Por último, se han realizado algunos experimentos con el objetivo de analizar la capacidad de adaptación del sistema a estas preferencias del usuario.

Abstract

This project describes Valencia Tourism Planner (VTP), an application that uses a recommender system and an automated planner to generate automatic and customized routes in Valencia. The personalization occurs at two stages: first, the recommender system obtains a list of activities according to the user profile and then, the user can specify some preferences with respect to the configuration of the agenda. This document describes in detail the design of the application and how the planning problem is built in order to take into account the user preferences. Finally, some experiments have been performed with the aim of analyzing the adaptability of the system to these user preferences.

Agradecimientos

En primer lugar agradecer al grupo de investigación Planning at King's del king's College de Londres la elaboración de OPTIC, el único planificador que implementa preferencias en el lenguaje PDDL y sin el cual no hubiese sido posible realizar este proyecto.

Además agradecer la posibilidad que me dieron Eva Onaindia y Laura Sebastiá para realizar este proyecto. Agradecer también la ayuda y confianza depositada en mí.

En último lugar agradecer a mi familia todo el apoyo tanto emocional como económico que me han dado y sin el cual no hubiese podido llegar hasta aquí.

Índice general

Capítulo 1

Introducción, motivación y objetivos

1.1. Introducción

En esta sección se explica la motivación del trabajo y se detallan los objetivos del mismo.

1.2. Motivación

Gran parte de la economía de España se basa en el sector turismo. Este sector atrae a multitud de gente tanto del mismo país, como de la Comunidad Europea y extracomunitarios.

La figura 1.1 muestra unos datos publicados por el INE (Instituto Nacional de Estadística) donde se puede observar que entre los años 2008 y 2012 se obtuvieron unos ingresos de más de 100 millones en el sector turístico, lo que corresponde a más de un 10 % del PIB de España. Estos datos demuestran la necesidad de cuidar y promover el sector turístico para que la gente pueda disfrutar del mismo y mantener o aumentar los ingresos obtenidos.

Por otro lado, existen multitud de técnicas de Inteligencia Artificial que pueden aplicarse al sector del turismo. En el presente trabajo se han utilizado sistemas de recomendación y técnicas de planificación inteligente o planificación automática para la generación de rutas turísticas por la ciudad de Valencia.

1.3. Objetivos

A continuación se exponen los objetivos que se pretenden cumplir con este proyecto:

- Especificación de una tarea de planificación a partir de los datos de entrada personalizados del usuario y de sus preferencias sobre la ruta.

Aportación del turismo al PIB de la economía española

	2008	2009	2010(P)	2011(P)	2012(A)
Precios corrientes: Millones de euros					
Total	113.628	105.355	108.599	112.908	112.035
Precios corrientes: Porcentaje sobre el PIB					
Total	10,4	10,1	10,4	10,8	10,9

Notas:
1) (P) : Estimación provisional
(A) : Estimación avance

Fuente: Instituto Nacional de Estadística

Figura 1.1: Datos económicos del turismo en España 2008 - 20012

- Recuperación de la información de la ciudad de Valencia mediante los servicios que proporciona Google. Por ejemplo, horarios de apertura y cierre de los lugares, cálculo de una ruta óptima entre dos lugares, etc.
- Generación automática del dominio y problema de planificación en lenguaje PDDL a partir de los datos del usuario y la información recuperada de la ciudad de Valencia.
- Conversión del plan devuelto por el planificador en una salida gráfica amigable y de fácil comprensión para para el usuario. Se mostrarán los lugares a visitar, la hora de comienzo de la visita y la duración aconsejada. Los desplazamientos se mostrarán en un mapa indicando las calles por las que el usuario debe desplazarse. El medio de transporte deseado habrá sido escogido por el usuario con anterioridad.

Estos son los objetivos inicialmente planteados en el proyecto Valencia Tourism Plannning y que se han cumplido durante la investigación realizada y el desarrollo del trabajo.

Capítulo 2

Estado del arte

2.1. Introducción

A continuación se presenta un breve recorrido por los sistemas de recomendación y su funcionamiento. Se hace especial hincapié en los Sistemas de Recomendación orientados al Turismo.

Seguidamente se resume brevemente el lenguaje de planificación PDDL, en particular las funcionalidades introducidas en las versiones PDDL 2.1, 2.2 y 3.0. Finalmente se ofrece una explicación de la fórmula empleada para estimar la distancia entres dos puntos GPS.

2.2. Sistemas de Recomendación

Un Sistema de Recomendación (SR) [27] es un servicio que actúa como filtro de información adaptativo, es decir, muestra solo la información relevante filtrando el resto de datos. Una ventaja de los SR es la personalización [14]. Personalización es la adaptación de la información a los intereses del usuario sin necesidad de que el propio usuario lo especifique explícitamente. La personalización funciona gracias a que el SR dispone de unos datos iniciales de los usuarios.

Un Sistema de Recomendación, por tanto, personaliza la información disponible o que se puede obtener filtrándola para conseguir que se adapte mejor a las necesidades, gustos o preferencias del usuario. Este proceso de personalización evita de este modo mostrar información que el SR infiere como no relevante o de particular interés para el usuario.

El funcionamiento de un SR se puede resumir en tres etapas [14]:

- **Recopilación de información:** Consiste en recopilar información sobre cada uno de los usuarios y almacenarla de forma que sea fácilmente entendible por el sistema.
- **Recomendación de elementos:** Antes de recomendar elementos, el SR aplica un proceso de razonamiento para inferir los ítems a recomendar. Y es precisamente en este razonamiento donde se aplica un filtrado de la información y se hace un razonamiento adaptado al usuario a partir de los datos de entrada del mismo.

El sistema se encarga de filtrar la información basándose en el conocimiento que se ha obtenido sobre los usuarios. Esta información filtrada llegará al usuario directa o indirectamente.

- **Grado de satisfacción:** Se mide el grado de satisfacción del usuario con los ítems recomendados. Esta tercera fase, permite al sistema mejorar los resultados de la recomendación obteniendo más información del usuario gracias a la retroalimentación (feedback) proporcionada por el mismo.

El origen de los SR se remonta a los años 90 con la aparición de Tapestry [18]. Tapestry es un SR basado en el filtrado colaborativo, este tipo de filtrado se explica en la sección 2.2.2. Técnicas de recomendación básicas. A día de hoy, multitud de sistemas web incorporan técnicas de recomendación entre los que se puede encontrar, Amazon y Google.

La utilización de SR puede aportar grades ventajas para la empresa y los usuarios. Principalmente, el gran beneficio que reporta la utilización de los SR es ofrecer una recomendación personalizada de acuerdo a la información y características del usuario en particular.

2.2.1. Perfil de usuario

Los SR parten del perfil de un usuario para calcular las recomendaciones. Un perfil de usuario contiene datos relevantes sobre los usuarios, de modo que cuantos más datos se tengan más acertada será la recomendación resultante. El perfil de usuario debe contener un mínimo de información sobre el mismo a fin de poder proporcionar recomendaciones significativas para el usuario.

El perfil de usuario tiene tres componentes principales:

- **Datos demográficos:** Datos personales del usuario, especialmente los relativos a su grupo demográfico.
- **Modelo de las preferencias del usuario:** Contiene las descripciones de los tipos de ítems o características generales preferidas por el usuario. En el caso de trabajar con grandes volúmenes de datos, estas preferencias se utilizan para determinar el subconjunto de ítems relevantes para el usuario. Esta transformación facilita la tarea del SR puesto que tiene que trabajar con menos datos. Habitualmente, para cada una de las características o preferencias del usuario se almacena una puntuación que viene dada por la valoración del usuario.
- **Histórico de interacciones:** El SR almacena un historial de peticiones e ítems recomendados en cada petición.

2.2.2. Técnicas de recomendación básicas

Un SR puede trabajar con diferentes técnicas de recomendación básicas que requieren los siguientes elementos [6]:

- **Base de datos inicial:** datos que existen antes de iniciar el proceso de recomendación. Entre estos datos se encuentran los ítems candidatos a ser recomendados por el sistema además de información especifica dependiente del algoritmo de recomendación empleado.
- **Datos de entrada:** información que el usuario proporciona al sistema para obtener una recomendación adecuada. Esta información puede ser tanto información personal como las preferencias contenidas en el perfil de usuario que ayuden a mejorar la recomendación.
- **Algoritmo recomendador:** algoritmo que combina la información de la base de datos y los datos de entrada del usuario para generar unas recomendaciones personales para un usuario.

Algunas de las técnicas de recomendación básicas existentes son las siguientes:

- **Recomendación demográfica:** [11] Los usuarios se clasifican en un grupo demográfico de acuerdo a sus datos personales. y la recomendación obtenida dependerá del grupo demográfico al que se pertenece. Como principal ventaja, nos encontramos ante uno de los métodos más sencillos además de que no necesita que el usuario puntúe características generales. El gran inconveniente de este sistema es la poca precisión de la recomendación obtenida.
- **Recomendación colaborativa:** [28] [11] Esta técnica obtiene una lista de ítems recomendados basándose en los gustos del usuario. La base fundamental sobre la que se sostiene es que usuarios similares tienen gustos parecidos. Uno de los principales problemas que tiene la recomendación colaborativa es que requiere un gran número de usuarios para un correcto funcionamiento. Una gran ventaja es que trabaja bien independientemente de la complejidad de los datos. Una gran ventaja es que permite recomendar ítems nunca vistos por el usuario. Este sistemas es el más habitual de todos ellos.
- **Recomendación basada en contenido:** [28] Esta técnica se basa en asociar una serie de características a cada ítem recomendable y devolver al usuario una lista de ítems cuyas características sean similares a otros ítems ya valorados por el usuario.

Cada uno de estos algoritmos de recomendación tienen una serie de ventajas e inconvenientes. Para resolver esos inconvenientes se utilizan los algoritmos híbridos.

2.2.3. Técnicas de recomendación híbridas

Las técnicas básicas tienen diferentes inconvenientes que limitan la calidad de los resultados obtenidos. Para paliar esta deficiencia, una solución son las técnicas híbridas [6], la base de estos sistemas híbridos en la combinación de varias técnicas básicas. Generalmente, una técnica híbrida consiste en la combinación de dos técnicas básicas. A continuación se muestran ejemplos de estos modelos mixtos [29]:

- **Recomendación ponderada:** Se realizan dos o más recomendaciones básicas y se combinadas linealmente. Un ejemplo bastante utilizado es la combinación de la técnica colaborativa y una técnica basada en contenido.
- **Recomendación mezclada:** Se realiza un procesado conjunto de más de una técnica. El resultado es una lista de elementos recomendados por los dos algoritmos.
- **Recomendador alternado:** El sistema recomendador usa una técnica u otra de forma alternativa. La decisión de que técnica elegir recae en algún criterio que ha de ser definido.
- **Recomendador en cascada:** Se ejecuta un recomendador detrás de otro, es decir, la salida de un recomendador es la entrada del siguiente. En este caso, el resultado obtenido depende del orden de ejecución.
- **Recomendador de combinación de características:** En primer lugar se usa una técnica de recomendación básica. El resultado de esta técnica se incluye como una nueva característica. El segundo algoritmo de recomendación, genera una recomendación utilizando los datos iniciales y los datos obtenidos por el primer algoritmo de recomendación. Esta última recomendación es la recomendación devuelta por el sistema.
- **Recomendador meta-level:** Técnica mediante la cual el primer recomendador genera un nuevo modelo completo de las preferencias de usuario. Este nuevo modelo es usado por el segundo recomendador para obtener la recomendación final.

El orden de ejecución de los distintos algoritmos básicos de recomendación puede influir en el resultado. Por ejemplo en el caso de un algoritmo en cascada o meta-level el resultado obtenido depende del orden.

2.2.4. Evaluación de un SR

La tercera etapa o fase del sistema consiste en medir el grado de satisfacción con la recomendación obtenida. La calidad de una recomendación se evalúa tradicionalmente con dos medidas, precisión y *recall*, heredadas del campo de *Information Retrieval*. A continuación se describe estas dos medidas y otras adicionales para evaluar un SR [32].

Dado un conjunto de elementos, se dice que son *relevantes* R aquellos que son de interés para el usuario. Se denota con S el conjunto de elementos recomendados (seleccionados para el usuario).

- **Precisión:** Mide el ratio de elementos recomendados que son relevantes.

$$precision = \frac{|S \cap R|}{|S|}$$

- **Recall:** Mide el ratio de elementos relevantes que han sido recomendados.

$$recall = \frac{|S \cap R|}{|R|}$$

- **Métrica F1:** Ambas medidas suelen usarse de forma combinada, en una medida denominada F1, es decir, es una combinación de la precisión y el *recall*. Normalmente, un mejor resultado en la precisión, conlleva un peor resultado en el *recall* y viceversa. Por ello, con F1, se obtiene una medida única que nos permite realizar comparaciones entre diferentes algoritmos de una forma más sencilla. F1 es una media armónica entre ambos valores:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Existe otro conjunto de medidas que se aplican a SR que muestran una lista ordenada de elementos recomendados. Así, se tiene en cuenta, no solamente cuántos elementos relevantes han sido seleccionados, sino también la posición en la lista en la que aparecen, ya que un elemento relevante es más útil si aparece antes en la lista ordenada. Este tipo de medidas (denominadas *Rank Score*) se definen, en general, como el ratio entre la posición del conjunto de elementos seleccionados con respecto a la mejor posición teórica que podrían haber alcanzado teniendo en cuenta las preferencias del usuario.

2.2.5. Aplicación al turismo

En esta sección se muestran varios sistemas de recomendación que contienen también una parte de planificación para el campo del turismo.

La gran mayoría de los sistemas existentes a día de hoy se basan en algoritmos híbridos que combinan técnicas básicas como la demográfica, la basada en contenido o la colaborativa [22]. Debido a que estos sistemas se crean para promover el turismo en una región concreta suelen limitarse a una ciudad o región, por ejemplo, SigTur/e-destination es un planificador de este estilo [2].

SAMAP [7] es otro sistema de recomendación híbrido que combina el uso de la técnica basada en contenido con la técnica colaborativa. Este sistema tiene la capacidad de analizar conexiones entre las actividades con distintos medios de transporte. Finalmente se puede obtener un fichero con la geoposición de los lugares y una explicación del plan. La figura 2.1 muestra la interfaz de la aplicación.

Otium [30] (figura 2.2), es un planificador turístico que utiliza un algoritmo basado en el contenido para buscar las actividades a realizar. Una vez generada la recomendación el sistema muestra una ruta junto con los horarios, que el usuario podrá modificar y finalmente descargarse.

Cada vez existen más sistemas con interfaz móvil, esto es debido al aumento del uso de teléfonos móviles inteligentes y a la conexión a Internet que poseen la gran mayoría de

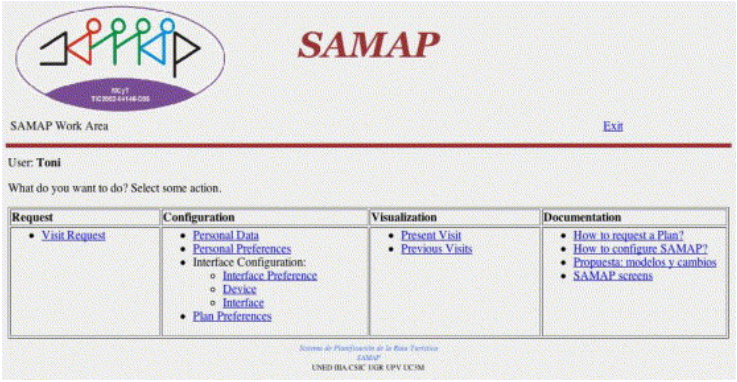

Figura 2.1: Muestra aplicación SAMAP

Figura 2.2: Muestra aplicación Otium

ellos. Dos ejemplos de sistemas con interfaz móvil son: CT-Planner y GeOasis [31]. Este último sistema, se comporta de forma similar a un guía turístico, mostrando descripción de los lugares además de guiar al usuario gracias al GPS integrado en los smartphones.

CT-Planner [23] [24] es un SR basado en contenido que permite una alta interacción del usuario con la aplicación. El sistema muestra varios planes y aprende de las preferencias del turista. Las preferencias que introduce el turista son duración de visitas, hora de inicio o día de la semana.

2.3. Planificación

El campo de la planificación en Inteligencia Artificial tiene como objetivo construir algoritmos de control que permitan a un agente sintetizar una secuencia de acciones que

le lleve a alcanzar sus objetivos. Un problema de planificación en IA es un problema de búsqueda que requiere encontrar una secuencia eficiente de acciones para conducir a un sistema desde un estado inicial hasta un estado objetivo.

Los problemas de planificación se suelen plantear en sistemas dinámicos donde, dado el estado actual y los objetivos, deducir la siguiente acción a aplicar no es una tarea obvia. La planificación es una tarea compleja y ésta es la razón por la cual la mayoría de planificadores trabajan sobre un modelo restringido del entorno (**planificación clásica**), siendo dicho modelo determinista, estático y totalmente observable. Concretamente, la planificación clásica fija las siguientes asunciones:

- **Modelo observable**. El mundo es totalmente observable y no existe información desconocida para el planificador.
- **Modelo estático**. Se puede predecir correctamente la evolución de las secuencias de acciones que se aplican sobre un estado inicial completamente conocido, ya que no hay influencias externas que afecten al entorno. Los objetivos son conocidos antes de comenzar la planificación y no cambian durante el transcurso del proceso de planificación.
- **Modelo determinista**. Los efectos de la aplicación de una acción en un estado son totalmente predecibles, lo que conduce determinísticamente a un único estado.
- **Enfoque proposicional**. Todas las variables del modelo de planificación (sentencias atómicas) pertenecen al dominio lógico, tomando por tanto los valores *cierto* o *falso*.
- **Acciones instantáneas**. Todas las acciones del modelo de planificación tienen la misma duración y se consideran atómicas e instantáneas.
- **Acciones no descomponibles**. Las acciones del modelo son directamente ejecutables en el entorno y no pueden descomponerse o dividirse en subacciones.
- **Planificación *offline***. La tarea de planificación consiste en construir un plan completo que satisface el objetivo antes de la ejecución de cualquier parte del mismo.

2.3.1. Planificación temporal

El principal problema de la planificación clásica es que las acciones no tienen duración, es decir, su ejecución es instantánea. Este tipo de modelización no permite representar la gran mayoría de problemas del mundo real, ni especificar restricciones, condiciones o preferencias temporales. Gracias a los grandes avances en planificación automática, actualmente se pueden modelar y resolver problemas más realistas, en donde la gestión del tiempo se puede controlar de forma explícita. Estos avances han permitido generar y resolver problemas temporales en donde las acciones tienen definida una duración y las condiciones y efectos de las mismas se pueden expresar en diferentes puntos de tiempo.

El objetivo principal de un planificador temporal es obtener un plan con la menor duración (*makespan*) total posible. A diferencia de los planes secuenciales, un plan temporal

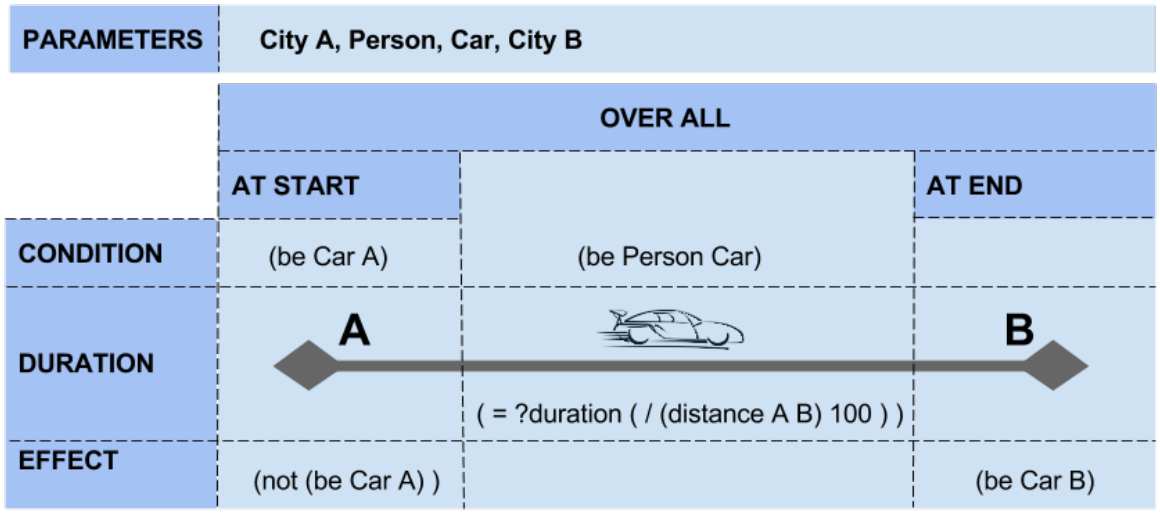

Figura 2.3: Ejemplo durative action

puede contemplar la realización de varias secuencias de acciones en paralelo. El objetivo, por tanto, no es minimizar el número de acciones sino el tiempo total o duración para lo cual hay que tener en cuenta el paralelismo entre las acciones.

El lenguaje estándar en planificación clásica es el lenguaje PDDL (Planning Domain Description Language), cuya primera versión fue PDDL1.2 [10]. Se trata de un lenguaje declarativo de predicados de primer orden que introduce los elementos necesarios para representar la descripción del dominio (jerarquía de objetos, predicados, operadores) y la descripción del problema (literales que describen el estado inicial y final del problema).

Los investigadores Fox y Long, actualmente en el King's College de Londres, desarrollaron la versión PDDL+ [12]. Tras una revisión de PDDL+, finalmente se propuso PDDL2.1 [13], que incluye todas las funcionalidades y elementos necesarios para definir problemas temporales y variables numéricas.

PDDL2.1 introduce las llamadas acciones durativas o acciones con duración que se especifican con el constructor `durative-action`. Un ejemplo de acción durativa se presenta en la figura 2.3 y su codificación en PDDL es la siguiente:

```
(:durative-action move
  :parameters (?x - city ?y - person ?z - car ?w - city)
  :duration
    (and (= ?duration (/ (distance A B) 100)))
  :condition
    (and
      (at start (be ?z ?x))
      (over all (be ?y ?z))
    )
  :effect
  (and
    (at start (not (be ?z ?x)))
```

```
    (at end (be ?z ?w))
  )
)
```

La acción `move` representa un desplazamiento llevado a cabo por una persona en un coche entre 2 ciudades, una de origen y otra de destino. La definición de la acción consta de tres partes:

- **Duración:** elemento nuevo con el que se define la duración de la acción. La duración de la acción de la Figura 2.3 se basa en la ecuación `tiempo = distancia / velocidad`. La aplicación de esta formula conlleva suponer que el vehículo lleva una velocidad media de 100 km/h y que la distancia entre las dos ciudades esta definida por la función `(distance A B)`. Finalmente la duración de la acción queda definida queda definido de la siguiente forma: `(= ?duration (/ (distance A B) 100))`. Existe también la posibilidad de definir la duración de las acciones como un valor en un intervalo temporal dejando al planificador la posibilidad de escoger el tiempo que considera mejor para la obtención del plan.
- **Condiciones:** contiene los predicados o condiciones que se han de cumplir para poder realizar la acción. Se distingue entre precondiciones que han de cumplirse al comienzo de la acción `(at start <predicate>)`, durante toda la acción `(over all <predicate>)` o al final de la acción `(at end <predicate>)`. En el ejemplo, al principio de la acción el vehículo se debe encontrar en la ciudad origen `(at start (be ?z ?x))`, y la persona debe estar en el vehículo durante toda la acción `(over all (be ?y ?z))`.
- **Efectos:** por último, hay que definir el resultado o efecto de la ejecución de la acción. Se distingue entre efectos que generan al inicio de la acción `(at start <predicate>)` o al final de la acción `(at end <predicate>)`. En el caso de estudio, el vehículo dejará de estar en la ciudad de origen al inicio de la acción `(at start (not (be ?z ?x)))`. Al borrar al inicio de la acción el literal que indica la posición del vehículo, esto evita solapar otra acción de movimiento que involucre al mismo coche. Por último, al final de la acción, el vehículo llegará a la ciudad destino `(at end (be ?z ?w))`.

La posibilidad de definir condiciones y efectos al principio y final de cada acción ofrece una mayor flexibilidad así como un mayor nivel de concurrencia o solapamiento entre las acciones del plan. Asimismo, la ejecución de una acción temporal en paralelo con otras o de forma secuencial dependerá sobre todo de que ninguna de las condiciones y efectos interfieren entre ellos o de que no se requiera un orden secuencial de las acciones. En el caso de estudio, un mismo vehículo y una misma persona no podrán desplazarse hacia dos ciudades distintas al mismo tiempo. El estudio de conflictos en planificación temporal es bastante más complejo que en planificación clásica por las múltiples interacciones que pueden surgir entre las acciones [1].

En el ejemplo anterior se ha utilizado la función `(distance A B)` para expresar la duración de la acción. Este es uno de los nuevos elementos que incorpora PDDL2.1 y que

consta de dos partes: funciones u operadores definidos en el fichero de dominio y *fluents* definidos en el fichero del problema. Las funciones representan problemas numéricos y los *fluents* el valor dado a cada problema numérico. La definición para este este ejemplo quedaría así: `(distance ?x - city ?y - city)` y en el fichero del problema se le asignaría un valor: `(= (distance A B) 50)`. En este caso, estamos declarando que la distancia entre la ciudad A y la B es de 50 km. Los *fluents* pueden formar parte de condición y efectos también.

La versión PDDL2.2 [9] se empleó en la cuarta competición internacional de planificación (IPC-4)[1]. PDDL2.2 incorpora una nueva funcionalidad que permite generar literales en un intervalo de tiempo determinado. Estos nuevos literales, activos durante un intervalo de tiempo se denominan Timed Initial Literal (TILs). Los TILs permiten activar o desactivar un hecho (verdadero o falso) en un instante de tiempo determinado. Los TIL se especifican en el estado inicial del problema y se corresponden, como todos los literales o fluents, con predicados definidos en el dominio del problema. La sintaxis correspondiente a la introducción de un TIL positivo es `(at <number><literal (name)>)` y, equivalentemente, poniendo el operador `not` delante para especificar un TIL negativo.

Imaginemos la situación anterior en donde un usuario se quiere desplazar de una ciudad a otra en un vehículo, pero el vehículo solo puede usarse de 10 de la mañana a 8 de la tarde. Para representar esta situación se genera un nuevo predicado `(free_car ?x - car)` y, además, se añade una nueva condición a la acción para indicar que el coche debe estar disponible `(at start (free_car ?z))`. Por último, en el fichero del problema se especifica el intervalo temporal en el que el coche esta disponible para usarse:

```
(:init
  [...]
  (at 10 (free_car car_A))
  (at 20 (not (free_car car_A))
  [...]
)
```

En el ejemplo se muestra que el objeto de tipo vehículo denominado `car_A` está disponible a partir de las 10:00 `(at 10 (free_car car_A))` y deja de estarlo a partir de las 20:00 `(at 20 (not (free_car car_A))`. En este ejemplo se considera que el instante de ejecución del plan es el instante t=0 y que la granularidad del tiempo se mide en horas.

2.3.2. Preferencias

Para la competición de planificación del 2006, se desarrolló la versión PDDL3.0 del lenguaje PDDL que incluía dos funcionalidades principales [16]:

- **Restricciones duras y blandas sobre la estructura del plan:** Esto incluye restricciones sobre la trayectoria de los estados por los que atraviesa la ejecución del plan y restricciones sobre la propia trayectoria del plan. Se trata, en definitiva, de

[1] http://www.icaps-conference.org/index.php/Main/Competitions

poder expresar restricciones sobre las acciones del plan y los estados intermedios del plan.

- **Objetivos duros y blandos:** Los primeros hacen referencia a objetivos que deben satisfacerse siempre tras la ejecución del plan y los segundos definen objetivos deseables pero no necesarios.

Las restricciones y objetivos blandos en PDDL3.0 se denominan preferencias. Son elementos definidos en el problema de planificación que se desea satisfacer para generar un "buen plan" aunque su cumplimiento no es necesario para obtener un plan correcto.

Existen varias formas de expresar y capturar la naturaleza de las preferencias, siendo las más habituales las aproximaciones cualitativas [17] y las aproximaciones que utilizan un esquema basado en recompensas [5]. En este trabajo nos centramos en las aproximaciones cuantitativas y las preferencias que se definen sobre objetivos del problema.

Hay dos aspectos relevantes a la hora de trabajar con preferencias: determinar cómo se van a definir e identificar las preferencias y determinar cómo se van a satisfacer, o, en su defecto, determinar el modo en el que el incumplimiento de una preferencia va a afectar a la calidad del plan. La semántica asociada a las preferencias es relativamente simple: en un plan que es correcto, las preferencias se satisfacen trivialmente en cualquier estado de dicho plan. Por otro lado, las preferencias se utilizan como una medida de la calidad del plan: un plan se considera de mejor calidad cuando satisface preferencias definidas por el usuario y de peor calidad en caso contrario. La dificultad surge a la hora de comparar dos planes que satisfacen un conjunto distinto de preferencias. En este caso, se utiliza una especificación del coste de no satisfacer una preferencia.

La sintaxis para definir una preferencia en PDDL3.0 es como sigue: `(preference [name] <GD>)` donde <GD>es un literal que representa un objetivo del problema.

```
(:goal
  [...]
  (preference p1 (be car_A city_b))
  [...]
)
```

En el ejemplo anterior se especifica que una preferencia del problema es que el coche denominado `car_A` esté en la ciudad `city_b` al finalizar el plan. En este caso, si al final del plan el vehículo `car_A` termina en `ciyt_b`, este plan será mejor que otro plan que no consiga `(be car city_b)`. en la ciudad B generaría un mejor plan que si no fuese así.

Las penalizaciones por el incumplimiento de una preferencia se especifican mediante la expresión `(is-violated <name>)`, donde `name` es un nombre asociado con una o varias preferencias. En PDDL3.0 no se puede especificar distintos grados de incumplimiento de las restricciones, únicamente se puede indicar el cumplimiento o no cumplimiento de la preferencia con la siguiente expresión:

```
(is-violated p1)
```

En PDDL los planes van asociados a un valor de una métrica, la cual se utiliza para guiar al planificador en la búsqueda de planes de buena calidad. La métrica se especifica

en PDDL como una función objetivo a minimizar. Particularmente, en lo que concierne a las preferencias, la penalización por el incumpliento de preferencias se incluye asimismo en la métrica que define la calidad del plan. Por ejemplo:

```
(:metric minimize
  ( +
    [...]
      (* 7 (is-violated p1))
    [...]
  )
)
```

En el ejemplo se especifica que el incumplimiento de la preferencia `p1 (be car_A city_b)`, referente a la posición final del coche `car_A` en la ciudad `city_b` conlleva una penalización de 7 unidades o puntos de coste. Si por el contrario, el vehículo se encuentra en `city_b` al finalizar la ejecución del plan, no existiría penalización.

Esto asegura que un plan que satisface más preferencias será de mejor calidad que un plan que satisface menos preferencias.

2.4. OPTIC

OPTIC [4] es un planificador de orden parcial que utiliza un proceso de búsqueda hacia adelante y que ha demostrado un excelente rendimiento en las competiciones de planificación, además de obtener planes de muy buena calidad (duración del plan). En la actualidad, OPTIC se puede considerar uno de los planificadores más relevantes en el estado del arte. La eficiencia de OPTIC se basa en una rápida generación de los nodos sucesores y en la utilización de una heurística independiente del dominio que es muy eficiente.

OPTIC aplica un proceso de búsqueda hacia delante en un espacio de planes de orden parcial. OPTIC hace uso del concepto de estado frontera que es el estado resultante de la simulación de la ejecución del plan del nodo del árbol de búsqueda. OPTIC utiliza el estado frontera para dos tareas:

- Para guiar el proceso de búsqueda seleccionando el nodo a expandir. El estado frontera de cada nodo permite aplicar heurísticas basadas en estados que son más informativas que las heurísticas basadas en planes.
- En el proceso de expansión de un nodo para generar nodos sucesores. OPTIC utiliza el estado frontera para determinar el conjunto de acciones que se pueden añadir al nodo a expandir y generar así los nodos sucesores.

En OPTIC, las acciones que se insertan en un nodo son únicamente aquellas cuyas precondiciones se satisfacen en el estado frontera. Este procedimiento ignora la posibilidad de insertar una acción cuyas precondiciones no se satisfagan en el estado frontera del nodo, aun cuando estas precondiciones se pudieran cumplir en algún punto intermedio del plan. De este modo, el proceso de generación de sucesores de OPTIC es un proceso incompleto

y para garantizar la completitud de la búsqueda, OPTIC recurre a un mecanismo de backtracking.

OPTIC soporta una buena parte del nivel 5 de PDDL2.1, que incluye acciones con efectos numéricos continuos (lineales) y efectos dependientes de las duraciones de las acciones. También soporta restricciones que no sean de obligado cumplimiento soft constraints y preferencias sobre los objetivos del problemas.

A diferencia de otros planificadores, OPTIC no trabaja con dos procesos independientes de selección de acciones temporales y ordenación de dichas acciones (scheduling), obteniendo planes de alta calidad con respecto a la duración. OPTIC es una versión extendida de POPF2 [8], que obtuvo el segundo lugar ex-aequo en la sección satisfacibilidad temporal de la IPC-7 del año 2011.

2.5. Estimación de distancias

La tarea del planificador temporal es calcular un plan con el conjunto de lugares a visitar con indicación de sus tiempo de comienzo y duración. Para tener en cuenta las restricciones y preferencias del usuario es necesario considerar los tiempos de desplazamiento entre dos visitas. Para ello, el planificador no calcula las distancias reales sino que realiza una estimación de las mismas. Esta estimación es muy fácil de resolver en un sistema cartesiano, mediante la fórmula Euclídea, pero mas complejo cuando se tiene una representación de la posición basada en el sistema de coordenadas GPS.

El sistema de coordenadas GPS permite geolocalizar un punto en la tierra. Para ello se necesitan tres coordenadas latitud, longitud y elevación. La latitud contiene la posición respecto al Ecuador o paralelo 0, considerando los extremos como 90^{o} norte o sur. La longitud es la posición Este u Oeste respecto al meridiano 0 o meridiano de Greenwich y puede oscilar entre 180^{o} este y 180^{o} oeste. La altitud es la diferencia de alturas desde un punto con respecto a otro y no es representativa para el caso de estudio ya que solo nos moveremos en la superficie terrestre. [25]

Existen varias formulas que permiten obtener o medir distancias entre coordenadas GPS. Muchas de ellas se basan en la suposición de que la tierra es redonda. Suponiendo que la tierra es redonda, se obtiene suficiente precisión para la mayoría de los casos. La fórmula escogida, por aportar una mayor precisión, es la fórmula Haversine [25]. Esta fórmula se basa en la trigonometría esférica para calcular la distancia en línea recta entre dos puntos.

La fórmula de Haversine o fórmula del semiverseno es un caso especial de una fórmula de trigonometría esférica que relaciona lados y ángulos de triángulos esféricos. En la figura 2.4 se observa el triángulo usado para el cálculo de la distancia, $distancia(v, w) = c$, es decir, el cálculo de la distancia en línea recta entre dos posiciones GPS v y w. La fórmula de Haversine es la siguiente:

$$d = 2 * R * \arcsin(\sqrt{hs(lat_v - lat_w) + \cos(lat_v)\cos(lat_w)hs(lng_w - lng_v)})$$

donde:

$$hs(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

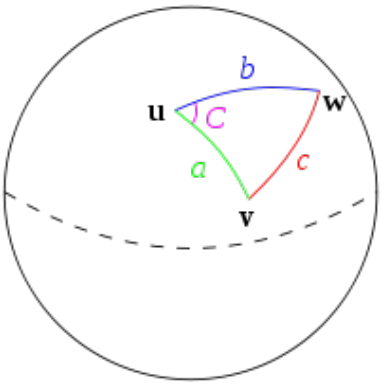

Figura 2.4: Explicación Haversine

La formula original de Haversine obtiene la distancia como una proporción, como un valor entre 0 y 1. Este valor obtenido ha de multiplicarse por el radio de la Tierra para la obtención de la estimación del valor. Esto conlleva la ventaja de que podría aplicarse a los demás astros con forma redonda o una aproximación a una esfera.

Capítulo 3

Diseño de Valencia Tourist Planner

3.1. Introducción

En este apartado se explica el diseño de la aplicación Valencia Tourist Planner y sus componentes principales. Se hace especial hincapié en la generación automática del dominio y problema de planificación que posteriormente ejecutará el planificador OPTIC para devolver un plan.

En primer lugar se muestra una breve introducción al diseño general de la aplicación junto con cada uno de los seis módulos que lo componen y su utilidad. A continuación se presenta una explicación sobre la aplicación web, en particular la estructura de la aplicación y la interfaz de usuario. Se presenta también los datos de carácter persistente necesarios para el funcionamiento de Valencia Tourist Planner (VTP), mostrando un esquema de la estructura interna en la que se almacena. Finalmente se muestra la planificación inteligente llevada a cabo con un detallado repaso por sus componente, centrando la atención en la función que se ha diseñado para obtener un plan o ruta turística adaptado al usuario.

3.2. Arquitectura de la aplicación

La aplicación se compone de varios módulos y servicios como se muestra en la figura 3.1. En el esquema de la figura 3.1 se observa que el nodo control es el módulo que coordina todo el sistema y al mismo tiempo de todos los demás módulos o componentes para funcionar. A continuación se realiza una breve introducción de cada uno de los módulos de la arquitectura y su papel en la aplicación VTP:

- **Servicios proporcionados por Google:** Google dispone de una serie de servicios públicos para el ámbito de la cartografía, geolocalización y generación de rutas. Su uso, siempre que no se supere una cuota asignada, conlleva un coste cero. El hecho de superar esta cuota bloquea el servicio u obliga a realizar una contribución económica. Cada uno de los servicios usados se explica a continuación de una forma más

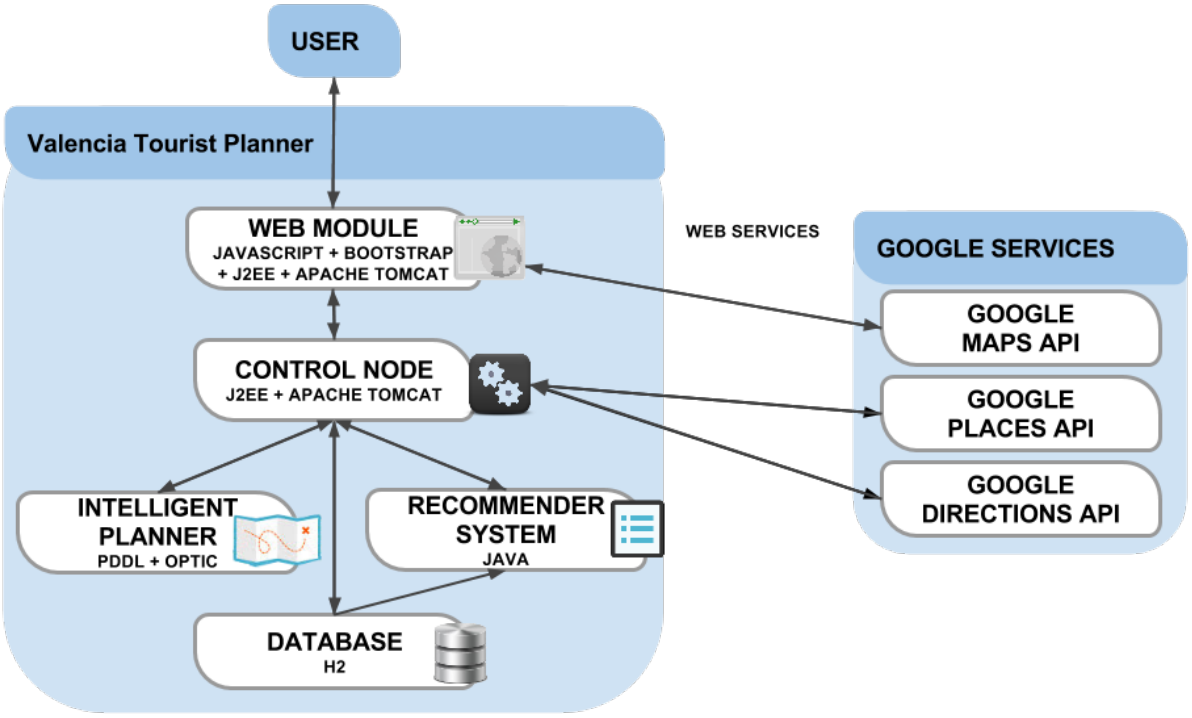

Figura 3.1: Esquema de la aplicación

detallada. Por último, Google también dispone de un servicio que permite localizar lugares (restaurantes, monumentos, estaciones, etc) y obtener detalles tales como valoraciones de los usuarios, posición GPS, horarios de apertura, etc.

- **Sistema de Recomendación:** aplicación desarrollada por el grupo GRPS [1] que devuelve una lista con lugares recomendados para ser visitados por el usuario. Esta lista depende de los gustos del usuario, de su historial y de la valoración dada por el usuario (retroalimentación). La lista contiene los lugares que se recomienda visitar al usuario así como un valor que indica el grado de la recomendación. El SR tiene que acceder a la base de datos, en donde consulta datos sobre el usuario y sobre la ciudad de Valencia para devolver la recomendación. Este módulo se ejecuta desde el nodo de control indicándole el identificador de usuario a recomendar.

- **Planificador Inteligente:** módulo encargado de generar el plan o ruta a seguir para el usuario. Este componente necesita dos ficheros, el que define el dominio de la aplicación (turismo) y el que define el problema (instancia para un usario particular), ambos escritos en PDDL y generados automáticamente por el nodo de control. Se ha empleado OPTIC como planificador temporal ya que es hasta la fecha el mejor planificador del estado del arte que maneja preferencias de usuario. OPTIC

[1] (http://users.dsic.upv.es/grupos/grps/)

devuelve un plan que no solo dependerá de la recomendación que devuelve el SR, sino también de los datos de la ciudad de Valencia y los parámetros iniciales que introduce el usuario.

- **Módulo Web:** este es el módulo web mediante el cual los usuarios acceden a la aplicación. Este componente se ha diseñado para funcionar en multitud de dispositivos independientemente del tamaño o potencia. Desde este componente se permite al usuario realizar todas las funciones disponibles y visualizar las rutas que devuelve el planificador. Se provee de una interfaz limpia, clara y sencilla en la que se muestran los datos que genera la aplicación.

- **Nodo de control:** el nodo de control y flujo de la aplicación es el encargado de gestionar y sincronizar todos los componentes de la aplicación. Este elemento solicita a cada servicio o componente la información que necesita en el momento apropiado. Además se encarga de codificar y generar el código PDDL en función de los datos de la ciudad, las recomendaciones dadas por el SR y las restricciones y preferencias del usuario.

- **Base de datos:** contiene todos los datos que utiliza la aplicación. Incluye datos de usuarios (datos personales, históricos y datos necesarios para el sistema de recomendación), información de rutas (almacena datos que permiten ahorrar consultas ya realizadas a las APIs de Google) y una lista de los lugares visitables de Valencia (latitud y longitud del lugar, detalles del mismo, etc).

La combinación de todos y cada uno de estos módulos ha permitido desarrollar la aplicación Valencia Tourist Planner. La calidad de los resultados dependerá en parte de la calidad de los datos obtenidos (las bases de datos de Google son actualizadas constantemente tanto por Google, como por usuarios y empresas) así como de la codificación del dominio y problema que posteriormente ejecutará el planificador para devolver un plan. A continuación se explica cada módulo más detalladamente.

3.3. Servicios proporcionados por Google

La aplicación hace uso de los servicios de Google Directions, Google Places y Google Maps. A la mayoría de estos servicios se accede mediante WebService como se explica a continuación:

- **Google Directions API WebService [19]:** API desarrollada por Google que mediante un protocolo HTTPS permite obtener un camino o ruta a seguir entre un origen y un destino. La gran ventaja de esta API es que funciona bajo coordenadas GPS, direcciones o nombres de los lugares. Además permite añadir puntos intermedios en las rutas y seleccionar el modo de desplazamiento (transporte público, andando, bici, etc). En este caso en concreto, Google Directions se ha utilizado para obtener un camino entre dos puntos y el tiempo necesario para realizar el desplazamiento entre dichos puntos. La comunicación entre este servicio y la aplicación se

realiza desde el nodo control que es el módulo que se encarga de enviar al servicio Google Directions los dos puntos entre los que se quiere obtener una ruta y el modo de desplazamiento. A continuación se muestra un ejemplo de petición:

```
https://maps.googleapis.com/maps/api/directions/json?origin=39.473671,
-0.364756&destination=39.465819,-0.381696&language=en&mode=driving&
key=API_KEY;
```

Con esta petición Google Directions busca una ruta entre el origen `origin=39.473671, -0.364756` y el destino `destination=39.465819,-0.381696` en coche `mode= driving`. Además se tiene que especificar el lenguaje en el que debe devolverse la información `language=en` y una clave que es diferente para cada aplicación `key=API_KEY`. Finalmente, Google ofrece la posibilidad de obtener los resultados a la petición en formato XML o formato JSON. Para la aplicación VCT se ha decidido utilizar JSON.

- **Google Places API Web Service [21]**: Servicio de Google que mediante petición HTTP permite obtener datos de un negocio, empresa o lugar. Los datos que se obtienen son de diferente tipo; por ejemplo, dirección del lugar, valoración del precio de la entrada, horarios de apertura u opiniones. Este web service se ha utilizado para obtener los horarios de apertura y cierre de cada lugar que puede ser visitado y para consultar los restaurantes disponibles en la zona. El nodo de control es el encargado de comunicarse con este servicio y en la mayoría de los casos realiza dos consultas por cada lugar: una primera para obtener el identificador Google del lugar, y una segunda para obtener los horarios de apertura y cierre para cualquier día de la semana. Para el caso de la localización de los restaurantes cercanos simplemente se debería realizar una petición al web sevice en la que se indica que se buscan restaurantes y la zona en la que se buscan.

- **Google Maps API [20]**: API desarrollado por Google que, al contrario de las anteriores, trabaja bajo lenguaje JavaScript. Esta API se invoca desde el módulo Web y se ejecuta en el ordenador cliente. Permite la visualización de mapas, ruta que se ha de seguir y muestra donde se encuentran todos y cada uno de los puntos a visitar por el usuario. Gracias a este servicio el usuario puede observar el recorrido de su ruta de forma visual. Este es el único servicio que se llama desde la aplicación web, puesto que pertenece a la capa de visualización de la información.

3.4. Módulo Web

El módulo Web está desarrollado con la tecnología J2EE y JavaScript. Su correcto funcionamiento se ha probado en servidores GlassFish y Apache Tomcat. La interfaz del módulo se ha creado utilizando HTML, CSS y el framework BootStrap. El módulo web se encarga de los siguientes cometidos:

- **Visualización de la información**: permite al usuario de la aplicación visualizar los datos de la futura ruta turística que va a realizar y obtener un historial de los

viajes ya realizados. La aplicación se puede visualizar en distintos dispositivos con diferentes características, ya que la interfaz se adapta a las pantallas y la mayoría del cálculo se realiza desde otros módulos.

- **Entorno amigable para el usuario**: dado que no todos los usuarios tienen un mismo nivel informático, se ha diseñado una aplicación de uso sencillo facilitando que cualquier usuario pueda llevar a cabo el proceso de planificación de rutas.
- **Personalización del resultado**: permite al usuario una personalización de la ruta en función a las necesidades/intereses del viaje en concreto. Esta personalización finalmente se traducirá y codificará en el lenguaje PDDL para que sea legible por el planificador OPTIC.

3.4.1. Interfaz Web

En esta sección se muestra el diseño de las interfaces del módulo web. Además, se explica cada uno de sus elementos y el modo de empleo de los mismos. Todas las pantallas de la aplicación muestran el menú principal desde el cual se puede acceder a la configuración de la ruta, lista de rutas y lista de lugares visitados, además, permitirá cerrar sesión.

- **Acceso a la aplicación**: el acceso a la aplicación VTP tiene lugar a través de la pantalla que se muestra en la Figura 3.2, en donde el usuario debe introducir su nombre de usuario y su contraseña. En esta pantalla se mostrarán imágenes relacionadas con el turismo en Valencia, monumentos, fiestas, etc.
- **Configurar ruta**: en la pantalla que se muestra en la Figura 3.3, el usuario introduce los datos y preferencias para planificar la ruta. Los datos necesarios para calcular una ruta son:
  - **Día en el que se va a realizar la ruta:** este es un dato importante puesto que dependiendo del día de la semana en el que se va a realizar la ruta el horario de visita de los lugares puede cambiar. Esto se indica en el campo denominado *Date* de la pantalla de la Figura 3.3.
  - **Hora de inicio y fin de la ruta:** la ruta tendrá lugar entre un intervalo temporal que el usuario deberá especificar. Esto se indica en los dos campos denominados *Time* de la pantalla de la Figura 3.3.
  - **Hora de comida:** el usuario define un intervalo de tiempo que será destinará para comer. Esto se muestra en la sección *Select eat time* de la pantalla de la Figura 3.3.
  - **Punto de salida y llegada:** en el mapa que se muestra en la Figura 3.3 el usuario debe seleccionar el lugar desde el que se quiere iniciar la ruta y el lugar de finalización de la misma.
  - **Medio de transporte:** se puede seleccionar cinco posibles modos de desplazamiento por la ciudad: andando, en bici, en coche, transporte público o un modo mixto. El modo mixto es un híbrido entre desplazamiento andando y

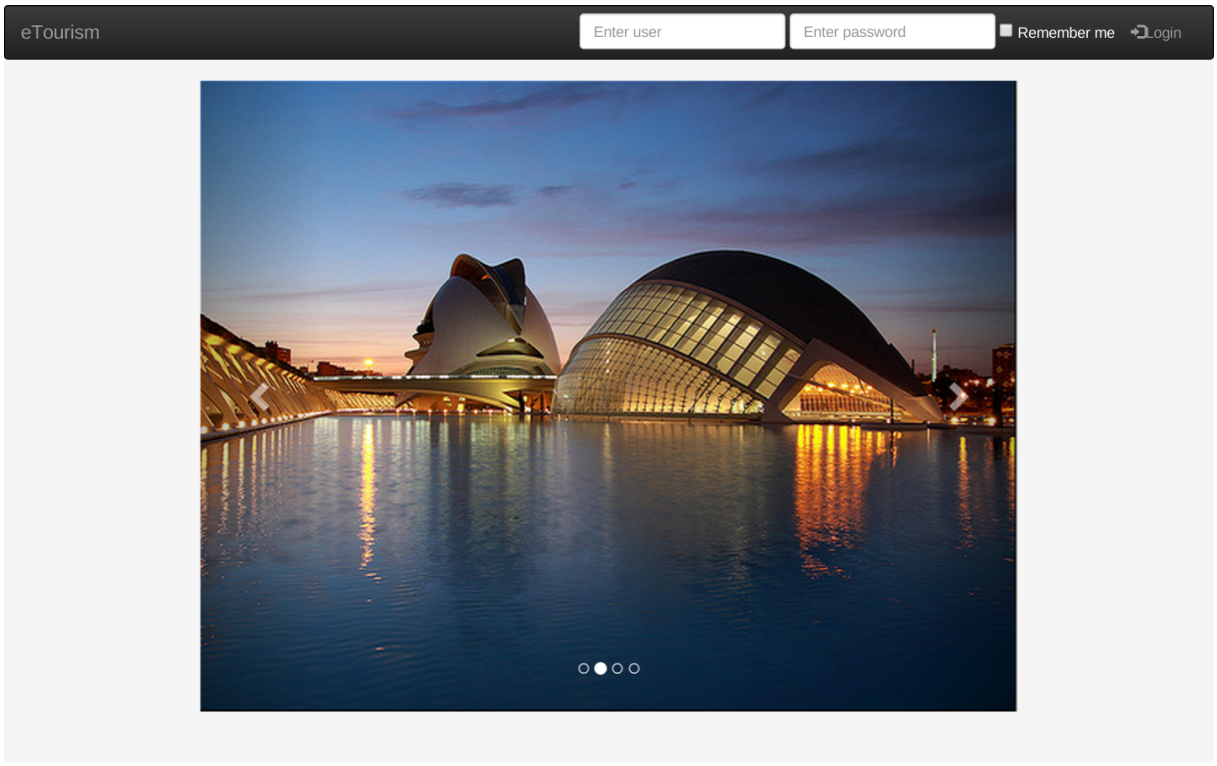

Figura 3.2: Interfaz - Acceso al Sistema (parte superior)

desplazamiento en transporte público. En este modo híbrido se combinan rutas a pie junto con rutas en transporte público y está pensado para que en los casos en los que el tiempo de desplazamiento sea demasiado grande no se obligue al usuario a andar y en los tiempos demasiado cortos no se obligue al usuario a buscar transporte público. Esto se indica en el campo denominado *Select mode* de la pantalla de la Figura 3.3.

El usuario debe indicar también sus preferencias respecto al tipo de ruta que desea realizar, con el objeto de conseguir un plan lo más personalizado y adaptado posible a las preferencias del usuario. Las preferencias a seleccionar son las siguientes:

- **Ocupación temporal:** el usuario selecciona el tipo de ruta: si desea realizar una ruta que ocupe el máximo tiempo posible o si prefiere una ruta más relajada, con más tiempos libres. Esto se indica en el campo denominado *Temporary occupation* de la pantalla de la Figura 3.3.
- **Número de lugares a visitar:** el usuario puede elegir entre tres posibles opciones: pocos lugares a visitar, muchos lugares o indiferente. Esto se indica en el campo denominado *Number of visit places* que se puede ver en la pantalla de la Figura 3.3.

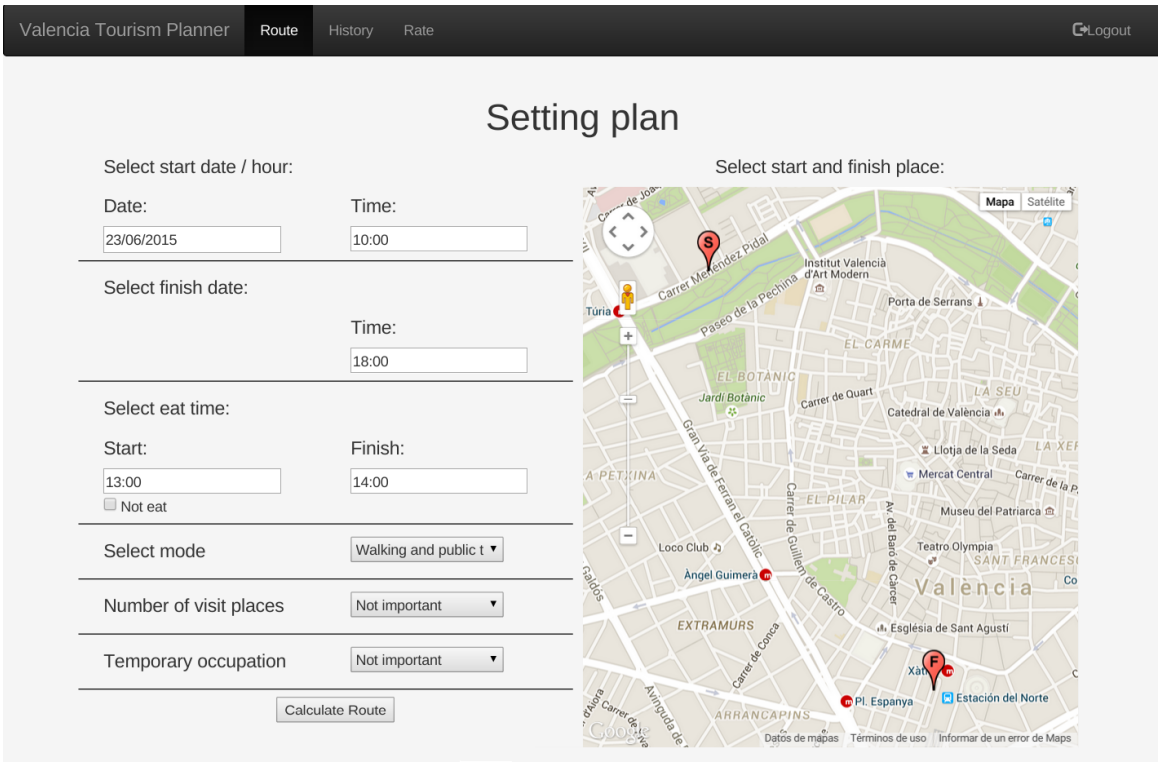

Figura 3.3: Interfaz - Preferencias de la ruta

- **Información de la ruta:** la Figura 3.4 muestra la planificación de una ruta, incluyendo los horarios de visita de cada lugar, el modo de desplazamiento para llegar a cada uno de los lugares y el tiempo requerido de desplazamiento.

- **Histórico de rutas:** pantalla que contiene una lista con las rutas ya realizadas (Figura 3.5) por el usuario. Desde esta pantalla el usuario puede acceder a una de las rutas ya realizadas.

- **Información de ruta pasada**: el acceso a esta pantalla se hace a través del enlace *Details* presente en la ventana que contiene la lista de rutas. Esta opción muestra al usuario la ruta turística que ha realizado y permite asignar una valoración a todos los sitios que ha visitado (Figura 3.6). La valoración que introduce el usuario (feedback) servirá al SR para mejorar y devolver futuras recomendaciones más adaptadas a los gustos del usuario. Existe siete posibles puntuaciones:

  - 1 - Pésima.
  - 2 - Mejorable.
  - 3 - Normal.
  - 4 - Buena.

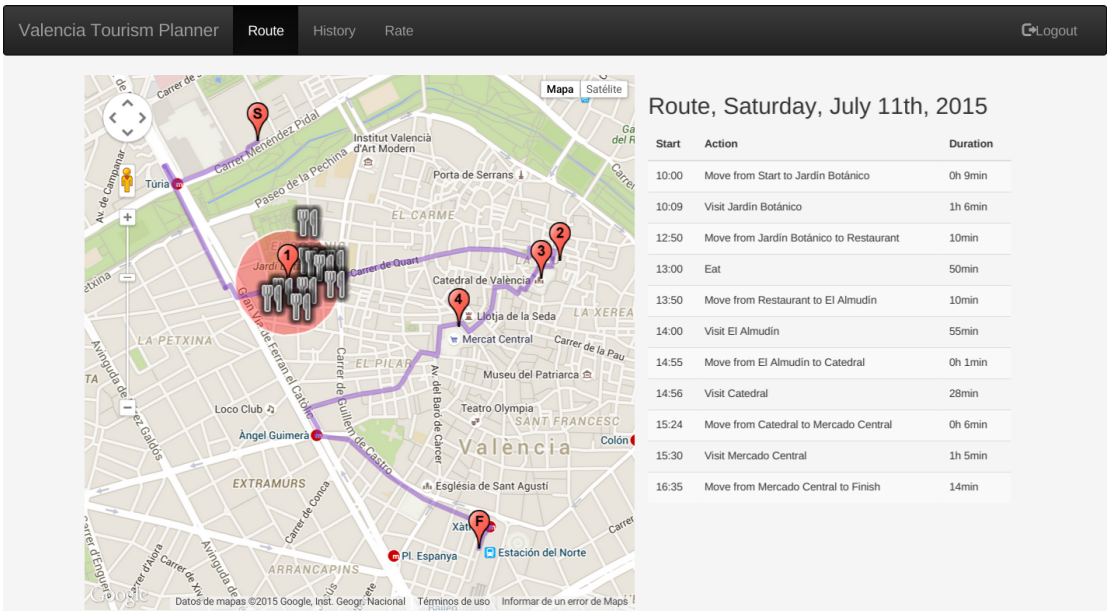

Figura 3.4: Interfaz - Mostrar ruta

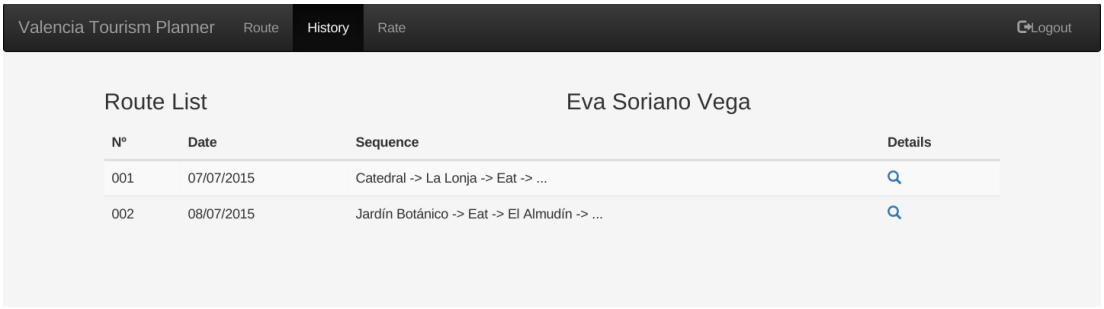

Figura 3.5: Interfaz - Histórico de rutas

- 5 - Impresionante.
- 6 - Volver a visitar.
- 7- No visitado

- **Lista de lugares visitados**: A esta ventana se accede desde la sección *rate* del menú principal. En esta ventana se muestra una lista de todos los lugares visitados por el usuario (figura 3.7), esta lista da la posibilidad al usuario de valorar todos los sitios que ya ha visitado con la ventaja de que se le muestran todos ellos en la misma lista sin tener que ir recorriendo todas las rutas ya realizadas.

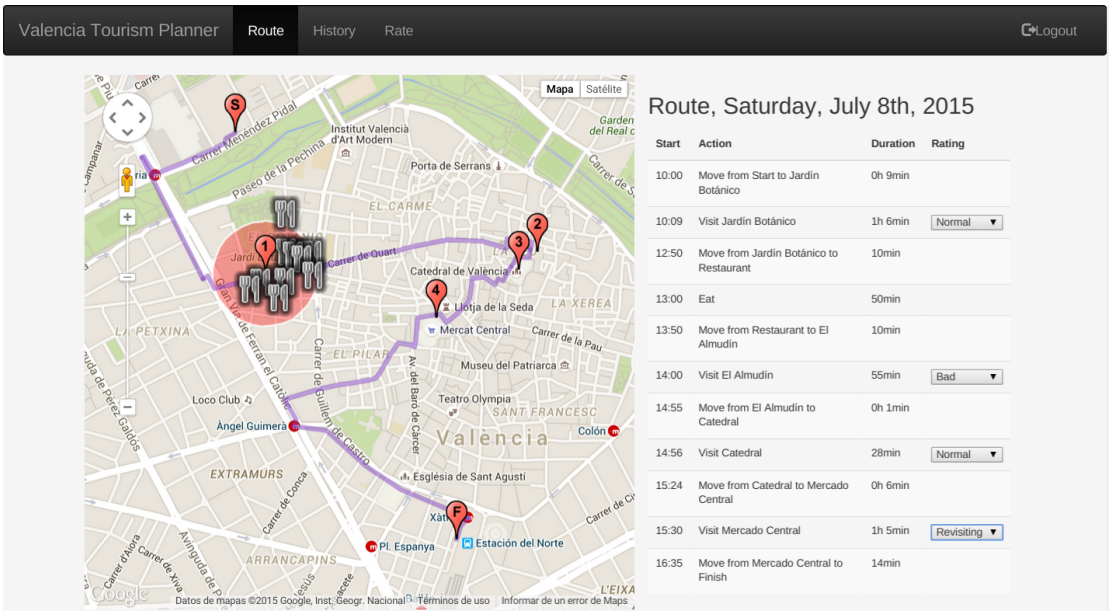

Figura 3.6: Interfaz - Ejemplo ruta ya realizada

3.5. Nodo de control

El nodo de control se ha implementado usando la tecnología Java Web, J2EE y se ha probado en servidores GlassFish y Apache Tomcat. El nodo de control es el módulo principal de la aplicación VCT, el cual se encarga de las siguientes tareas:

- **Control de flujo**: esta tarea es la encargada de comunicar todos los módulos y de solicitar su ejecución cuando es necesario. Las llamadas a los servicios se pueden realizar a través de peticiones HTTPS o lanzando directamente la ejecución del módulo o servicio necesario. Finalmente, el nodo control recibe los datos que devuelve cada uno de los servicios.

- **Retroalimentación del sistema de recomendaciones:** una parte importante de la aplicación VCT es la retroalimentación del sistema. El usuario puede acceder al historial de rutas desde el cual puede valorar cada uno de los lugares visitados y mejorar así la calidad de las rutas. Además garantiza que no se repitan lugares ya visitados.

- **Generación dinámica de código PDDL:** la aplicación genera automáticamente el fichero PDDL del dominio y del problema. La generación de estos dos ficheros depende de varios factores:

    - **Resultado del recomendador:** el factor principal es el resultado del recomendador que devuelve la lista de sitios recomendados para el usuario.

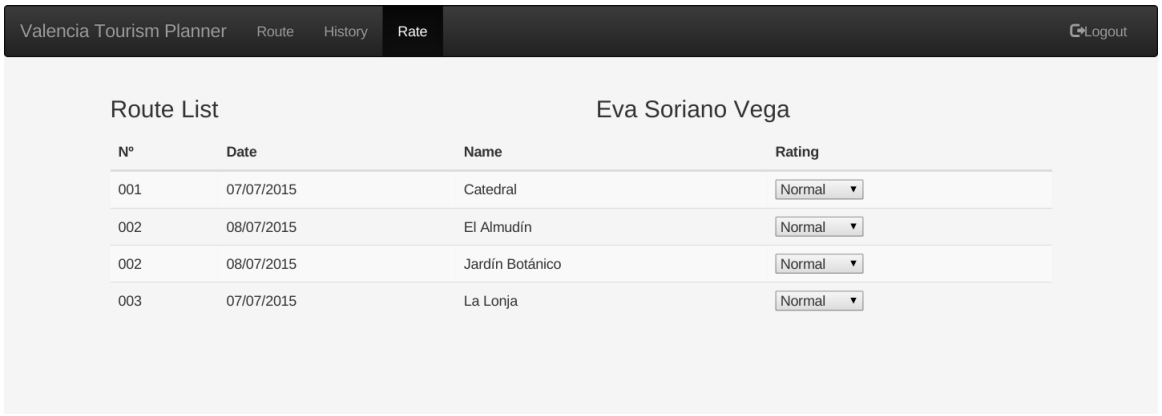

Figura 3.7: Interfaz - Lista de lugares visitados

- **Horarios de cada uno de los lugares:** se comprueba también los horarios de apertura y cierre de cada uno de los sitios la fecha escogida para la realización de la ruta. Estos horarios se almacenan en la base de datos reduciendo el número de peticiones al servicio de Google Places. Cada cierto tiempo, la base de datos se actualiza para mejorar las rutas generadas.
- **Estimación de tiempos de desplazamiento:** la aplicación estima el tiempo mediante la formula *Haversine* y la fórmula que permite obtener el tiempo sabiendo la velocidad media y la superficie recorrida. A esta fórmula hay que sumarle una corrección por tiempos muertos y por desplazamiento a través de las calles puesto que la estimación es en línea recta. La velocidad empleada en la formula depende del vehículo que se emplee para el desplazamiento: .

$$EstimatedTime = \frac{distance}{speed} + \theta * distance$$

  donde $distance = HaversineDistance(A, B)$ y $speed$ depende del vehículo.
- **Preferencias del usuario:** Cada vez que se planifica una ruta, el usuario introduce una serie de parámetros que el sistema codificará en lenguaje PDDL. Estos parámetros se han detallado en la sección 3.4.1. entre los que se encuentra, por ejemplo, el número de lugares a visitar o la ocupación temporal.

- **Convertir el plan:** El planificador devuelve el plan en un fichero de texto que el nodo de control debe recuperar y extraer el mejor plan.

- **Obtener la ruta:** una vez el planificador devuelve el plan, se accede al servicio Google Directions para obtener los caminos óptimos de los desplazamientos entre los lugares a visitar devueltos por el planificador.

3.6. Base de datos

Esta sección describe la base de datos que se utiliza en VTP y que contiene toda la información necesaria para el proceso de recomendación y planificación de las rutas.

El sistema gestor de base de datos utilizado es H2 [2], una base de datos SQL muy rápida, de reducido tamaño, fácil de migrar a otros sistemas y permite, además, el acceso en local o en servidor. El diseño de la aplicación VTP se ha hecho permitiendo solo al nodo de control realizar operaciones de inserciones y modificaciones, el sistema de recomendación también accede a la base de datos pero solo realizar operaciones de consulta.

A continuación se muestra un esquema general de los elementos presentes en la base de datos:

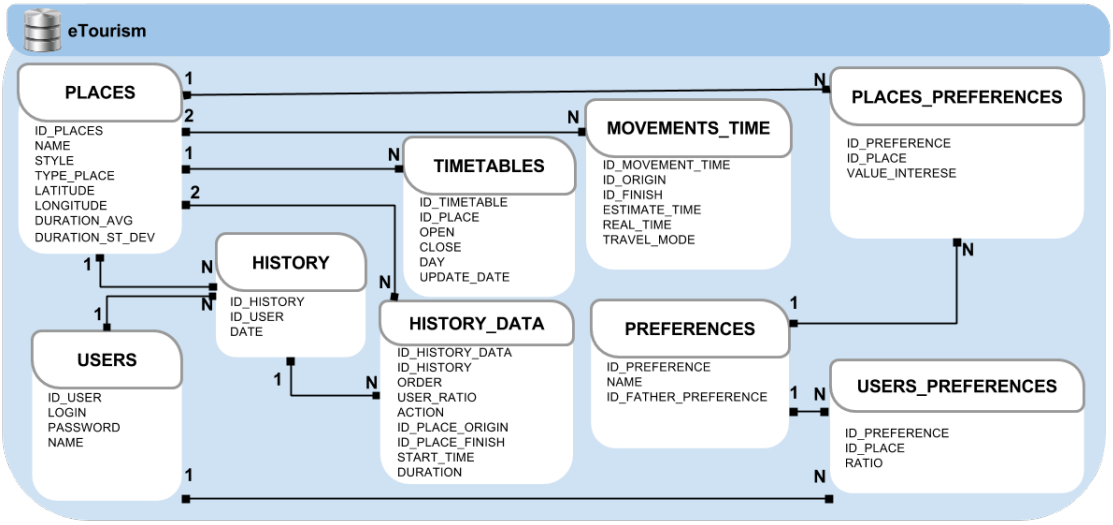

Figura 3.8: Esquema lógico de la Base de Datos

Se procede a explicar brevemente cada una de las tablas que componen la base de datos:

- **Places:** Contiene toda la información relativa a los lugares que pueden ser visitados, su ubicación, nombre y estilo.
- **Users:** Contiene información básica de los usuarios que se han registrado en el sistema.
- **History:** Almacena un historial de los datos generales de las rutas planificadas con anterioridad y que han sido llevadas a cabo por cada uno de los usuarios.
- **History_Data:** Contiene cada uno de los pasos o acciones propuestos en una ruta planificada para un determinado usuario. Para cada acción, se almacena su instante

[2] http://www.h2database.com/html/main.html

de comienzo y su duración. Además, en el caso de que se trate de una acción de desplazamiento, se almacena el punto de origen y el punto de destino. Por otro lado, con el fin de mejorar la calidad de las recomendaciones ofrecidas al usuario, cuando éste accede de nuevo a la aplicación, se le solicita que valore las rutas recomendadas anteriormente. Esta valoración (por lugar) se almacena también en esta tabla y será utilizada a posteriori por el recomendador para mejorar el perfil del usuario.

- **Timetables**: Almacena información sobre los horarios y días de apertura y cierre de todos los lugares visitables. Esta información será actualizada con cierta periodicidad para evitar problemas por desactualización.

- **Movements_time**: Almacena información relativa a las rutas y tiempos de desplazamiento de un origen a un destino. Esta información será actualizada cada cierto tiempo para obtener mejores y más precisas rutas.

- **Preferences:** La tabla Preferences contiene una lista de características que serán utilizadas en la fase de recomendación. Estas características (o preferencias) servirán tanto para describir los lugares a recomendar como para describir el perfil del usuario. Concretamente, esta tabla se relaciona con la tabla Places (a través de la tabla Places_Preferences), de forma que se describe cada uno de los lugares a recomendar por medio de un conjunto de características/preferencias junto con un valor que indica el nivel de adecuación de dicho lugar con una determinada característica. También se relaciona esta tabla con la tabla Users (a través de la tabla Users_Preferences), con el fin de recoger las preferencias estimadas de los usuarios por parte del recomendador junto con un valor entre 0 y 100 que indica una estimación del grado de interés del usuario en dicha preferencia.

3.7. Sistema Recomendador

Como se ha explicado anteriormente, VTP consta de varios módulos, entre los que se encuentra el sistema de recomendación (SR). Particularmente, el SR utilizado en esta aplicación es GRSK (*Generalist Recommender System Kernel*) [15], desarrollado en el grupo GPRS de la UPV. Este SR es un sistema de recomendación generalista, es decir, puede ser utilizado con diversos dominios, siempre y cuando éstos estén descritos por medio de una ontología (en el caso de VTP, la ontología se encuentra representada en la tabla Preferences de la base de datos). GRSK utiliza una técnica de recomendación híbrida mezclada, que puede combinar (en función de las necesidades de la aplicación) las siguientes técnicas básicas de recomendación: demográfica, basada en contenido, colaborativa y basada en preferencias actuales.

Así, en el marco de VTP, GRSK recibe el identificador del usuario para el que se debe realizar la recomendación. En este caso, se utilizan las técnicas basada en contenido y colaborativa como técnicas básicas. Con la información disponible del perfil del usuario, GRSK proporciona al nodo central una lista de las actividades recomendadas para dicho usuario, junto con un grado de interés estimado para cada una de estas actividades. Es

decir, el nodo central recibirá una lista ordenada de tuplas de la forma: $\langle a, Pr^a \rangle$, donde a denota la actividad recomendada y $Pr^a \in [0, 300]$ denota el grado de interés estimado para dicho usuario en la actividad a.

Sobre ese sistema original se ha realizado una extensión que permite, no sólo obtener la lista de actividades recomendadas, sino también una estimación del intervalo de duración de la visita, que depende del grado de interés estimado para el usuario.

Imaginemos dos usuarios diferentes que utilizan VTP para planificar su visita a Valencia, de forma que GRSK determina que ambos usuarios deberían visitar el *Museo de las Ciencias*, pero con un valor diferente del grado de interés estimado. Esto implica que el *Museo de las Ciencias* debería aparecer en ambas agendas, pero probablemente el tiempo que cada usuario debería emplear en dicha visita debería ser diferente. Por tanto, esta extensión de GRSK permite que, al mismo tiempo que se realiza la recomendación de actividades, se recomienda un intervalo de duración de la visita, que se ajustará de acuerdo al grado de interés estimado para el usuario, a partir de la duración media de la actividad.

Así, se ha asignado, para cada lugar, una duración media μ de la visita, que indica la duración recomendada de la visita para un turista típico. Esta duración μ, junto con una varianza δ^2, definen una distribución normal $X(\mu, \delta^2)$ que GRSK utiliza para recomendar el intervalo de duración de la visita. Actualmente, estos valores han sido introducidos manualmente para cada lugar, pero como trabajo futuro, nuestra idea es estimar esta distribución mediante el estudio del comportamiento real de los turistas a través de un análisis de sus interacciones en Twitter, similar al que se realiza en [3]. Una vez la distribución normal $X(\mu, \delta^2)$ para cada visita ha sido definida, el intervalo de duración recomendado se calcula como:

$$min^a_{dur} = X(Pr^a/300/2)$$

$$max^a_{dur} = X(Pr^a/300)$$

Por ejemplo, asumamos que el *Museo de las Ciencias* tiene una duración media $\mu = 180$ y $\delta = 36$. De acuerdo con la definición de la distribución normal, esto implica que el 68 % de los turistas pasan 180+36=216 minutos en esta visita, 95 % de los turistas pasan 180+36*2=252 minutos y, finalmente, el 99 % pasan 180+36*3=288 minutos. Si GRSK determina un grado de interés estimado para un usuario de 100 sobre 300, el intervalo de duración recomendado será $[145, 164]$, mientras que si el grado de interés es 260, el intervalo de duración será $[174, 220]$.

Una vez este intervalo de duración recomendado se ha calculado, GRSK retorna un conjunto de tuplas de la forma: $\langle a, Pr^a, dur^a_{min}, dur^a_{max} \rangle$. A continuación se muestra un ejemplo de la salida del SR:

3.8. Codificación tarea de planificación

El módulo de la codificación tarea de planificación se encarga de generar una ruta turística para el usuario que incluya los lugares a visitar que se encuentran entre sus prefe-

rencias y que satisfagas sus restricciones temporales. Para ello se hace uso del planificador OPTIC, el cual recibe dos ficheros PDDL y devolverá la ruta recomendada al usuario.

3.8.1. Dominio de planificación: turismo

El fichero de dominio contiene la especificación de los tipos de objetos, predicados y operadores para un entorno de turismo. El fichero de dominio describe los elementos necesarios para la especificación de un entorno genérico de turismo por lo que la descripción de este dominio serviría para generar rutas en cualquier ciudad y para cualquier usuario, ya que la información de ciudad y usuario se detalla en el fichero de problema. Los tipos de objetos que se emplean para describir el dominio de turismo son:

```
(:types
  person
  visitable restaurant hotel - location
)
```

Estos tipos representan los tipos de los objetos que describen el problema a resolver. `person` es el tipo para representar el usuario que quiere realizar la ruta, se utiliza para tener información de donde se encuentra la persona en cada momento. Se incluye además tres tipos `visitable, restaurant` y `hotel` de tipo `location`: un lugar `visitable` es un lugar turístico que puede ser visitado por el usuario para el que se genera el plan. Un `restaurant` será cualquier sitio en el que el usuario pueda comer, y por último, `hotel` es el tipo con el que el sistema hace referencia al punto de origen y fin de la ruta, que pueden ser el mismo punto o no.

Los predicados que se utilizan para describir la información de los estados del problema son lo siguientes:

```
(:predicates
    (be ?x - person  ?y - location)
    (visit_location ?x - location)
    (not_visit_location ?x - location)
    (eaten)
    (not_eaten)
    (open ?x - location)
    (active)
    (can_move)
  )
```

- **be**: predicado con el que se expresa el lugar (`location`) en el que se encuentra la persona en cada instante y tiene la siguiente propiedad:

$$\forall a, b \in location \wedge c \in person \quad / \quad be(c, a) \Rightarrow \nexists be(c, b) \wedge a \neq b$$

- **visit_location**: este predicado está activado para cada lugar visitable cuando el lugar ya ha sido visitado. Al inicio del plan, todos los lugares aparecen como no visitados y el planificador buscará los lugares que mejor se adapten a las preferencias del usuario.

- **not_visit_location**: este predicado expresa la información opuesta a `visit_location`. La razón de incluirlo es que OPTIC no admite precondiciones negadas en la definición de los operadores. Estos dos predicados se utilizan para evitar que un mismo lugar se visite más de una vez.

$$\forall x \in location \quad / \quad visit\_location(x) = \neg not\_visit\_location(x)$$

$$\wedge not\_visit\_location(x) = \neg visit\_location(x)$$

- **eaten:** predicado con el que se especifica si la persona ha comido o no.
- **not_eaten:** este predicado es necesario debido a que OPTIC no maneja precondiciones negadas en los operadores. Su significado es opuesto al anterior, es decir:

$$eaten = \neg not\_eaten \wedge not\_eaten = \neg eaten$$

- **open:** predicado que representa si un lugar esta abierto o no. Los horarios de apertura de los lugares se especifican mediante TILs, tal y como se detalló en el capítulo 2 en la explicación de PDDL2.2. [4]. Un lugar ha de estar abierto para poder visitarse, en caso contrario no podrá realizarse la visita. Para obtener información sobre si un lugar está abierto o no se consulta la API Google Places y se codifica para que el planificador lo entienda.
- **active:** este predicado se utiliza para generar en el fichero de problema un TIL que represente la ventana temporal definida por el usuario para la realización de la ruta. Este predicado se utiliza para controlar que el planificador no excede los límites temporales marcados por el usuario.
- **can_move:** predicado que controla si un usuario puede desplazarse de un lugar a otro o no. Su principal función es reducir el tiempo de búsqueda del plan puesto que evita la realización de movimientos sin sentido. Este predicado se activa cuando se realiza cualquier acción excepto desplazarse y se desactiva cuando se desplaza.

Las funciones que se han definido para el dominio de turismo son:

```
(:functions
    (location_time ?x - location ?y - location)
    (min_visit_time ?y - location)
    (max_visit_time ?y - location)
    (eat_time)
    (total_available_time)
    (number_visit_location)
    (transport_time)
  )
```

Las funciones mostradas anteriormente se utilizan para:

- **location_time**: devuelve el tiempo de desplazamiento entre dos localizaciones de tipo `location`. Este tiempo vendrá dado en minutos y será una estimación calculada mediante la fórmula explicada en la sección 3.5 (*EstimatedTime*).

- **min_visit_location**: devuelve la duración mínima recomendado para la visita de un lugar (tipo `location`). Si el lugar `?y` se incluye en la ruta del usuario, la duración de esta visita nunca será inferior a `min_visit_location`. El tiempo, al igual que la función `location_time`, se expresa en minutos.
- **max_visit_location**: devuelve la duración máxima recomendado para la visita de un lugar (tipo `location`). Si el lugar `?y` se incluye en la ruta del usuario, la duración de esta visita nunca será superior a `min_visit_location`. El tiempo, al igual que la función `location_time`, se expresa en minutos.
- **eat_time**: devuelve el tiempo destinado para comer. Variará de un plan a otro y de la persona en cuestión puesto que es el mismo usuario el que lo introduce. El tiempo se expresa igualmente en minutos.
- **total_available_time**: devuelve el tiempo máximo disponible para realizar la visita en la fecha seleccionada. El valor de esta función se decrementará a medida que el plan vaya incluyendo visitas a lugar y los correspondientes desplazamientos. Tiempo expresado en minutos y dependerá de la fecha y hora introducidas por el usuario para iniciar y finalizar la ruta.
- **number_visit_location**: almacena el número de lugares visitados, inicialmente toma siempre el valor cero. Será usado para obtener una solución que se adapte lo mejor posible a las preferencias del usuario.
- **transport_time**: función que almacena el tiempo consumido en el desplazamiento entre los lugares a visitar que se incluyen en el plan. El tiempo se expresa en minutos y se incrementa por cada desplazamiento del plan.

Por último, se muestran los operadores que definen el dominio de turismo. Los operadores, tal y como se describió en la sección 2.3.1, se componen de un conjunto de condiciones y efectos que se requieren o generan al comienzo, al final o durante toda la ejecución de la acción. Las condiciones y efectos se expresan mediante los predicados y funciones descritas en esta sección. A continuación se muestra algunos ejemplos de operadores y se procede a explicar el funcionamiento de todos ellos. Para ver el código completo del dominio, ver anexo 1 y anexo 2.

```
(:durative-action move
  :parameters (?x - location ?y - person ?z - location)
  :duration (= ?duration (location_time ?x ?z))
  :condition
    (and
      (at start (can_move))
      (over all (active))
      (at start(be ?y ?x))
      (at start (>= (total_available_time) (location_time ?x ?z)))
    )
  :effect
    (and
      (at start(not (be ?y ?x))) (at end(be ?y ?z))
```

```
      (at end (decrease (total_available_time) (location_time ?x ?z)))
      (at end (increase (transport_time) (location_time ?x ?z)))
      (at end (not (can_move)))
    )
)
```

- **move:** acción que permite mover una persona de un lugar a otro. Las características principales esta función son:
  - **condiciones necesarias:** en primer lugar el tiempo para la realización del plan no debe haber sido sobrepasado, es decir, `(active)` tiene que estar activo. Además, para llevar a cabo esta acción debe estar activo el predicado `(can_move)`, el cual se utiliza para evita realizar dos acciones de desplazamiento consecutivas. Por otro lado, la persona debe estar en el lugar de origen desde el que se desea mover `(be ?y ?x)` y debe disponer de tiempo suficiente para realizar la acción `(>= (total_available_time) (location _time ?x ?z))`.
  - **duración**: la duración de la acción se supone fija `(= ?duration (location _time ?x ?z))`. Inicialmente, esta distancia se estima con la fórmula explicada en la sección 3.5 y una vez ya encontrado el plan se solicitará a Google el tiempo y ruta precisos para obtener el plan con mucha mayor precisión.
  - **efectos**: al iniciar la acción el usuario ya no estará situado en la localización de origen `(not (be ?y ?x))`, y una vez acabada la acción el usuario estará situado en la localización de destino `(be ?y ?z)`, se reduce el tiempo disponible para realizar otras acciones `(decrease (total_available_time) (location _time ?x ?z))` y el tiempo de transporte aumenta `(increase (transport_time) (location_time ?x ?z))`. Finalmente se desactiva el predicado que permite realizar una acción `move (not (can_move))`.

```
(:durative-action visit
  :parameters (?x - location ?y - person)
  :duration
    (and
      (>= ?duration (min_visit_time ?x))
      (<= ?duration (max_visit_time ?x))
      (<= ?duration (total_available_time))
    )
  :condition
    (and
      (over all (active))
      (at start (not_visit_location ?x))
      (over all (be ?y ?x)) (over all (open ?x))
    )
    :effect
      (and
        (at end (can_move))
        (at start (not (not_visit_location ?x)))
        (at end (visit_location ?x))
        (at end (decrease (total_available_time) ?duration))
```

```
        (at end (increase (number_visit_location) 1))
        (at end (increase (visit_overtime) (- ?duration (min_visit_time ?x))))
      )
)
```

- **visit**: acción a través de la cual un usuario visita un visitable.
  - **condiciones necesarias**: en primer lugar, como en todas las acciones, se comprueba que estemos en el plazo de ejecución de acciones `(active)`, el lugar no ha de haber sido visitado con anterioridad `(not_visit_location ?x)`, el lugar debe estar abierto `(open ?x)` y se debe de estar en el lugar concreto `(be ?y ?x)`. Si una de estas condiciones no se cumple la acción no puede ser llevada a cabo.
  - **duración**: la duración de la acción es variable oscilando entre los dos tiempos dados por el recomendador para este visitable `(>= ?duration (min_visit_time ?x))` y `(<= ?duration (max_visit_time ?x))` y teniendo en cuenta que nunca puede sobrepasar el tiempo máximo disponible `(<= ?duration (total_available_time))`.
  - **efectos**: el usuario pasa a haber visitado el lugar `(visit_location ?x)` y `(not (not_visit_location ?x))`, se incrementa el número de lugares visitados `(increase (number_visit_location) 1)`, se decrementa el tiempo disponible para realizar más acciones `(decrease (total_available _time) ?duration)`. Finalmente se debe habilitar el desplazamiento, sino sería imposible encontrar una solución `(can_move)`.

```
(:durative-action eat
  [...]
)
```

- **eat**: acción mediante la cual un usuario come. Esta acción habilita un tiempo en el que el usuario debe comer. Aunque no es parte de esta sección se debe explicar que el sistema recomendará lugares en las cercanías al sitio en el que se encuentra para poder comer y dejará al usuario la elección del mismo.
  - **condiciones necesarias**: en primer lugar que este dentro del intervalo de tiempo que permite realizar acciones `(active)`, que no haya comido todavía `(not_eaten)`, se haya desplazado al restaurante `(be ?y ?x)`, se este dentro del plazo en el que se permite comer `(open ?x)` y se tenga tiempo suficiente para comer `(>= (total_available_time) (eat_time))` y que este dentro del intervalo de tiempo que el usuario ha definido para comer.
  - **duración**: la duración ha sido definida por el usuario con anterioridad `(= ?duration(eat_time))`.
  - **efectos**: los efectos producidos son que el usuario ha comido `(not (not_eaten))` y `(eaten ?y)`, permitir ejecutar la acción de desplazamiento otra vez `(can_move)` y el tiempo para realizar el plan disminuye `(decrease (total_available_time) (eat_time))`.

3.8.2. Problema de planificación: usuario

El fichero que contiene el problema es generado automáticamente por la aplicación web y contiene las preferencias introducidas por el usuario, los posibles lugares a visitar recomendados por el SR, y los horarios de apertura de los lugares que podría visitar.

En primer lugar se añaden los objetos con los que tiene que trabajar el problema:

```
(:objects
  P - person
  id_17380 id_17374 id_17396 id_17419 id_17431 id_17402 id_17375 - visitable
  id_55 - restaurant
  id_01 id_99 - hotel
)
```

- Se añade la persona que va a realizar la ruta `P - person:`.
- Todos los lugares propensos a visitarse que el SR ha recomendado para el usuario como por ejemplo `id_17380 - visitable`. El sistema maneja identificadores de lugares para trabajar, en el ejemplo se usa id_17380 identificador que corresponde al Museo de las Ciencias Príncipe Felipe.
- Restaurante ficticio que una vez planificada la ruta será sustituido por un conjunto de restaurantes cercanos al lugar donde el planificador ha propuesto que se coma y de los que el usuario podrá escoger el que más le guste `id_55 - restaurant`.
- Lugar de origen desde el que el usuario inicia la ruta `id_17380 - visitable`.
- Destino o final de ruta `id_17380 - visitable`.

A continuación se define el estado inicial del problema en el que se representan todos los valores de las funciones y predicados activos necesarios para la correcta resolución del problema:

```
(:init
  (be P id_01)

  (at 60  (open id_17380))
  (at 420 (not (open id_17380)))
  [...]
  (= (location_time id_01 id_17380) 71)
  [...]
  (= (min_visit_time id_17380) 170)
  [...]
  (= (max_visit_time id_17380) 220)
  [...]
  (not_visit_location id_17380)
  [...]
  (at 0 (active))
  (at 480 (not (active)))

  (= (eat_time) 50)
```

```
  (= (total_available_time) 480)
  (= (number_visit_location) 0)
  (= (transport_time) 0)
  (can_move)
  (not_eaten P)
)
```

Antes de proseguir con la explicación se debe entender que los valores temporales del problema están expresados en minutos, que se considera el minuto 0 como el minuto en el que se inicia la ruta y en función a ese valor se expresan los demás valores temporales.

- **Ubicación inicial:** el primer elemento definido en el estado inicial es la ubicación del usuario al inicio de la ruta `(be P id_01)`.
- **Horarios de apertura y cierre de los locales:** se definen los horarios de apertura y cierre de cada uno de los lugares que el usuario puede visitar, por ejemplo, `(at 60 (open id_17380)) (at 420 (not (open id_17380)))`, esto significa que el Museo Príncipe Felipe abre una hora más tarde del inicio de la ruta y cierra siete horas más tarde del inicio de la ruta también.
- **Estimación temporal de desplazamiento:** se expresa la estimación del desplazamiento entre todos los pares de `(location)`, por ejemplo `(= (location_time id_01 id_17380) 71)`.
- **Tiempos mínimos y máximos de cada visita:** el sistema añadirá el intervalo temporal que puede durar una visita, por ejemplo `(= (min_visit_time id_17380) 170) (= (max_visit_time id_17380) 220)`.
- **Visitables no visitados:** se debe incluir este predicado que significa que los visitables no han sido visitados al inicio de la ruta `visitable`.
- **Intervalo de ejecución de acciones:** la aplicación definirá el intervalo temporal entre el que las acciones pueden ser ejecutadas de la siguiente forma `(at 0 (active)) (at 480 (not (active)))`, en el ejemplo se muestra un plan que puede durar un máximo de 8 horas (480 minutos).
- **Tiempo de comida:** el tiempo que el usuario ha decidido dedicar a la comida se incluye en el fichero de problema de esta forma `(= (eat_time) 50)`.
- **Tiempo total disponible:** se inicializa el tiempo del que se dispone para realizar las acciones `(= (total_available_time) 480)`.
- **Número de lugares visitados:** se inicializa la función que controla el número de lugares que se han visitado `(= (number_visit_location) 0)`.
- **Tiempo dedicado al transporte:** es inicializada la función que controla el tiempo dedicado a la acción `move (= (transport_time) 0)`.

- **Permisión de desplazamiento:** se especifica que la primera acción a ejecutarse puede ser desplazarse `(can_move)`.
- **Ausencia de comida:** se especifica que el usuario no ha comido al inicio del problema `(not_eaten P)`.

3.8.3. Preferencias del usuario y función objetivo

Una vez definido el estado inicial del problema se definen los objetivos que se deben cumplir al final del problema tanto objetivos de obligado cumplimiento como preferencias.

```
(:goal
  (and
    (be P id_99)
    (eaten P)

    (preference p1 (visit_location id_17380))
    (preference p2 (visit_location id_17374))
    [...]
    (> (number_visit_location) 0)
  )
)
```

En este ejemplo se especifican tres objetivos de obligado cumplimiento:

- **(be P id_99):** el usuario debe finalizar la ruta en el punto definido por él en la introducción de datos.
- **(eaten P):** el usuario debe haber comido antes de finalizar el plan.
- **(> (number_visit_location) 0):** para evitar la posibilidad de que el planificador devuelva planes en los que no se realizan visitas se ha especificado que se debe realizar almenos una visita.

A continuación se especifican las preferencias, en el fichero del problema debe haber una preferencia por cada visitables obtenido del sistema recomendador, por ejemplo, `(preference p1 (visit_location id_17380))`.

La función objetivo es una función a minimizar donde se especifica el valor de penalización de cada predicado no cumplido y mediante la cual se buscará el mejor plan posible.

```
(:metric minimize
  (+
    (* (/ (* 260 (is-violated p1)) 1595) 100)
    (* (/ (* 255 (is-violated p2)) 1595) 100)
    (* (/ (* 240 (is-violated p3)) 1595) 100)
    (* (/ (* 230 (is-violated p4)) 1595) 100)
    (* (/ (* 210 (is-violated p5)) 1595) 100)
    (* (/ (* 200 (is-violated p6)) 1595) 100)
    (* (/ (* 200 (is-violated p7)) 1595) 100)
```

```
    (* (/ (transport_time) 480) 100)
    (* 0 (* (/ (number_visit_location) 7) 100))
    (* 0 (* (/ (total_available_time) 480) 100))
  )
)
```

Antes de explicar cada tipo de penalización, introducimos los siguientes conceptos: se define RA como el conjunto de visitables que devuelve el SR y Π como el conjunto de acciones incluidas en el plan. Por lo tanto Π_v son el conjunto de acciones `visitar`, Π_m son el conjunto de acciones `move` y Π_e es la acción `eat`. $T_{initial}$ corresponde con el intervalo de tiempo inicialmente disponible por el usuario. Finalmente se define T_{final} como el tiempo libre al final de realizar todas las acciones:

$$T_{final} = T_{initial} - \sum_{a \in \Pi} dur(a)$$

- **Penalización por lugares no visitados:** Como ya se ha explicado el recomendador devuelve una lista ordenada de lugares recomendados para cada usuario $\langle a, Pr^a \rangle$ donde a es la actividad recomendada y P la recomendación $Pr^a \in [0, 300]$. La recomendación será usado para penalizar el no visitar ese visitable.

  $$P_{not\_visited} = \frac{\sum_{a \in RA - \Pi} Pr^a}{\sum_{a \in RA} Pr^a}$$

  Ratio de la suma de las prioridades de los lugares no visitados entre la suma del total de prioridades.

  En el ejemplo mostrado al principio de la sección esta penalización quedaría de esta forma:

  ```
  (:metric minimize
    (+
      (* (/ (* 260 (is-violated p1)) 1595) 100)
      (* (/ (* 255 (is-violated p2)) 1595) 100)
      (* (/ (* 240 (is-violated p3)) 1595) 100)
      (* (/ (* 230 (is-violated p4)) 1595) 100)
      (* (/ (* 210 (is-violated p5)) 1595) 100)
      (* (/ (* 200 (is-violated p6)) 1595) 100)
      (* (/ (* 200 (is-violated p7)) 1595) 100)
      [...]
    )
  )
  ```

- **Penalización por tiempo de desplazamiento**: Penalización con la que la aplicación VTP busca reducir el tiempo de desplazamiento entre lugares, por lo tanto, busca crear una ruta eficiente entre los posibles lugares a visitar. La fórmula usada para implementarlo es la siguiente:

$$P_{move} = \frac{\sum_{a \in \Pi_m} dur(a)}{T_{initial}}$$

La fórmula se explica como la suma de los tiempos de las acciones `move` entre el total del tiempo del problema.

En el ejemplo mostrado al principio de la sección esta penalización quedaría de esta forma:

```
(:metric minimize
  (+
    [...]
    (* (/ (transport_time) 480) 100)
    [...]
  )
)
```

- **Penalización por número de lugares visitados**: El usuario tiene la posibilidad de especificarle a la aplicación que intente incrementar el número de lugares a visitar o que intente decrementarlo. Mediante esta penalización se puede observar una modificación en el número de lugares visitados, siempre teniendo en cuenta que existen más variables y que es posible que el resultado no sea suficientemente significativo.

$$P_{\#visits} = \begin{cases} (-1) * \frac{|\Pi_v|}{|RA|} & : \quad if \quad pref_{\#visits} = many \\ \frac{|\Pi_v|}{|RA|} & : \quad if \quad pref_{\#visits} = few \\ 0 & : \quad if \quad pref_{\#visits} = indiferent \end{cases}$$

Una forma sencilla de entender la formula es que se obtiene el ratio entre lugares visitados y lugares visitables. Si se quiere usar para maximizar el número de lugares se multiplicará por -1, si se quiere minimizar el número de lugares a visitar se multiplicará por 1 y si no se quiere usar se multiplicará por 0.

En el ejemplo mostrado al principio de la sección esta penalización quedaría de esta forma:

```
(:metric minimize
  (+
    [...]
    (* 0 (* (/ (number_visit_location) 7) 100))
    [...]
  )
)
```

- **Penalización por ocupación temporal**: Se encarga de controlar la ocupación temporal del sistema, dependiendo de la preferencia del usuario, se generarán planes con una ocupación temporal alta (poco o nada de tiempo libre entre las acciones del plan) o con una ocupación temporal baja (bastante tiempo libre a lo largo del plan).

$$P_{occupation} \begin{cases} \frac{T_{final}}{T_{initial}} & : \quad if \quad pref_{occupation} = high \\ 10 * \frac{T_{final}}{T_{initial}} & : \quad if \quad pref_{occupation} = low \\ 0 & : \quad if \quad pref_{occupation} = indiferent \end{cases}$$

  Ratio entre Tiempo disponible al final de realizar todas las acciones y tiempo total disponible. En caso de buscar baja ocupación temporal se multiplica por 10 y en caso de no darle importancia se multiplica por 0.

  En el ejemplo mostrado al principio de la sección esta penalización quedaría de esta forma:

```
(:metric minimize
  (+
    [...]
    (* 0 (* (/ (total_available_time) 480) 100))
  )
)
```

La función objetivo que guiará al planificador OPTIC en la búsqueda de una solución se obtiene mediante la suma de los cuatro tipos de penalizaciones, habiendo normalizado antes todas las penalizaciones a un intervalo entre 0 y 100.

$$P_{total} = (P_{not\_visited} + P_{move} + P_{\#visits} + P_{occupation}) * 100$$

Se ha de mencionar que la precisión de las formulas creadas no ha podido ser mayor debido a que OPTIC no permite realizar la división de dos funciones.

3.8.4. Ejemplos de aplicación de penalizaciones

A continuación se muestran una serie de experimentos realizados sobre cada una de las penalizaciones antes explicadas para comprobar su correcto funcionamiento.

En primer lugar se va a comprobar la **penalización por lugares no visitados**. Se ha creado un mapa sencillo (Figura 3.9) en que que hay tres visitables entre los cuales solo hay diferencia de prioridades:

Tras ejecutar el planificador se ha generado el plan que se muestra a continuación y en el que se observa como realmente si que visita los lugares con mayor prioridad (visitable1 y visitable2):

```
;;;; Solution Found
; States evaluated: 74
```

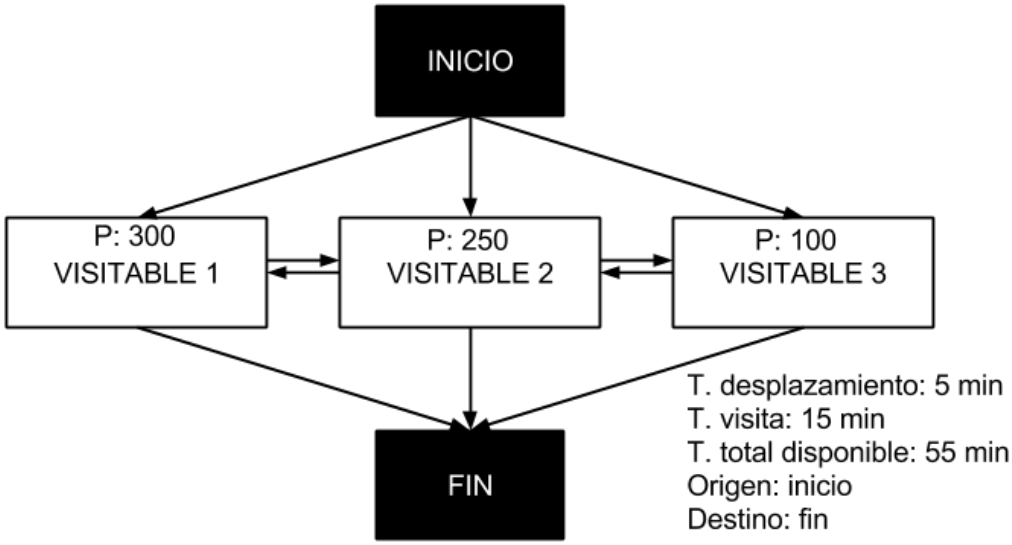

Figura 3.9: Ejemplo - máximizar la prioridad del visitable

```
; Cost: 7.692
; Time 0.40
0.002: (move origin p visitable2)  [5.000]
5.003: (visit visitable2 p)  [15.000]
20.004: (move visitable2 p visitable1)  [5.000]
25.005: (visit visitable1 p)  [15.000]
40.006: (move visitable1 p finish)  [5.000]
```

A continuación se va a comprobar el correcto funcionamiento de la **penalización por tiempo de desplazamiento** que nos permiten minimizar el tiempo de desplazamiento lo máximo posible, esta es una parte esencial para la correcta creación del plan ya que evita que se pierda tiempo en el desplazamiento. Se han creado el siguiente mapa sencillo (Figura 3.10) en el que hay dos posibles rutas que visitan los dos visitables pero una requiere mucho menos tiempo de desplazamiento que la otra.

El mejor resultado posible que el sistema puede devolver incluyendo los dos visitables y reduciendo el tiempo de desplazamiento lo máximo posible es el que se muestra en la Figura 3.11.

Finalmente el planificador genera el plan esperado como se muestra a continuación:

```
; Plan found with metric 26.667
; Theoretical reachable cost 26.667
; States evaluated so far: 44
; States pruned based on preheuristic cost lower bound: 0
; Time 0.22
0.000: (move origin p visitable1)  [10.000]
10.001: (visit visitable1 p)  [10.000]
20.002: (move visitable1 p visitable2)  [15.000]
35.003: (visit visitable2 p)  [10.000]
```

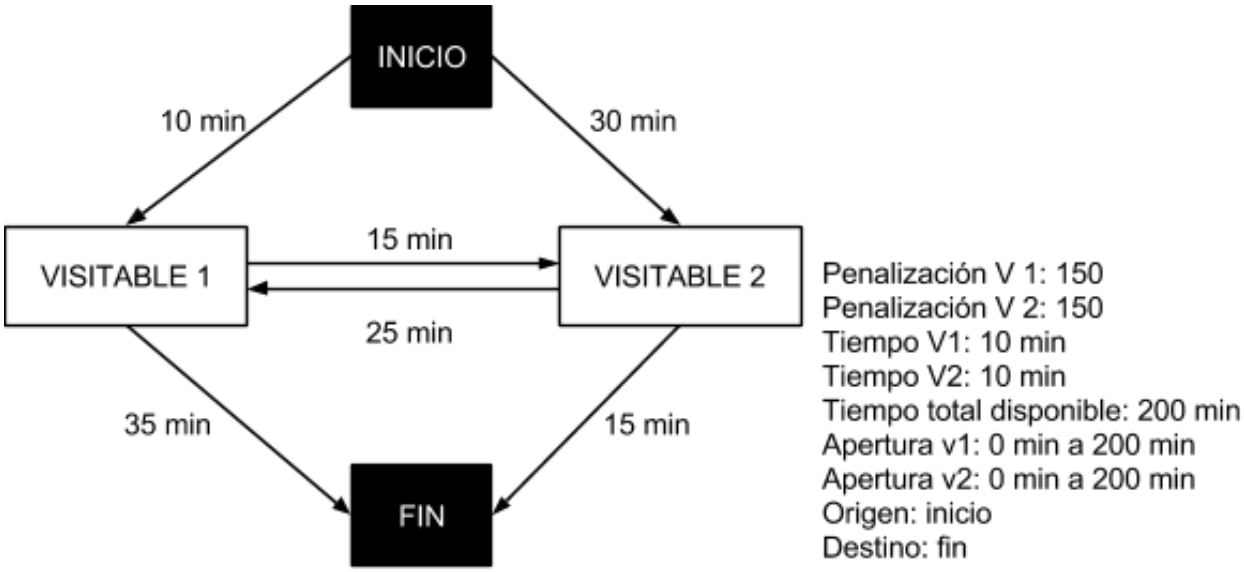

Figura 3.10: Ejemplo - minimizar tiempo de desplazamiento

Figura 3.11: Solución - minimizar tiempo de desplazamiento

```
45.004: (move visitable2 p finish)  [15.000]
```

A continuación se analiza la **penalización por número de lugares visitados** Para comprobar el correcto funcionamiento se ha generado un mapa con cuatro visitables, figura 3.12.

En el mapa se observa que existen dos posibles planes representativos:

- Visitar visitable1 y cualquier otro visitable más. El tiempo total disponible no permite visitar más lugares.
- Visitar los tres visitables con menor tiempo de visita, el tiempo no permitiría visitar el VISITABLE1.

En la primera prueba se busca minimizar el número de lugares a visitar y el resultado es el esperado, solo se visitan dos de los cuatro lugares disponibles:

```
; Plan found with metric 600.000
; Theoretical reachable cost 600.000
; States evaluated so far: 12
; States pruned based on preheuristic cost lower bound: 0
```

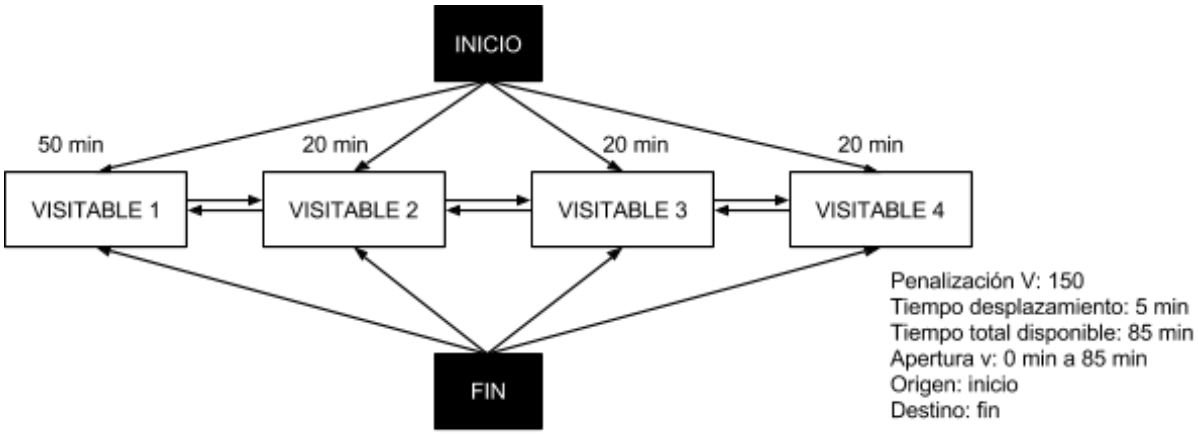

Figura 3.12: Ejemplo - cantidad de lugares a visitar

```
; Time 0.05
0.001: (move origin p visitable1)  [5.000]
5.002: (visit visitable1 p)  [50.000]
55.003: (move visitable1 p visitable4)  [5.000]
60.004: (visit visitable4 p)  [20.000]
80.005: (move visitable4 p finish)  [5.000]
```

A continuación se realiza la prueba de maximizar el número de lugares a visitar sobre el mismo mapa, y el resultado es el siguiente (visita tres visitables):

```
; Plan foung with metric 750.000
; States evaluated so far: 2004
; States pruned based on preheuristic cost lower bound: 0
; Time 11.00
0.001: (move origin p visitable4)  [5.000]
5.002: (visit visitable4 p)  [20.000]
25.003: (move visitable4 p visitable3)  [5.000]
30.005: (visit visitable3 p)  [20.000]
50.006: (move visitable3 p visitable2)  [5.000]
55.007: (visit visitable2 p)  [20.000]
75.008: (move visitable2 p finish)  [5.000]
```

Por último, se comprueba la penalización de la ocupación temporal del plan. Para ello se genera un mapa en el que hay tres lugares a visitar y cuyo tiempo de visita va entre 20 y 40 minutos (Figura 3.13).

En caso de buscar minimizar la ocupación temporal las visitas deberán tener un valor cercano a 20 min y eso es lo que genera el planificador:

```
; Plan found with metric 8000.000
; Theoretical reachable cost 14000.100
; States evaluated so far: 16
; States pruned bases on preheuristic cost lower bound: 0
; Time 0.08
```

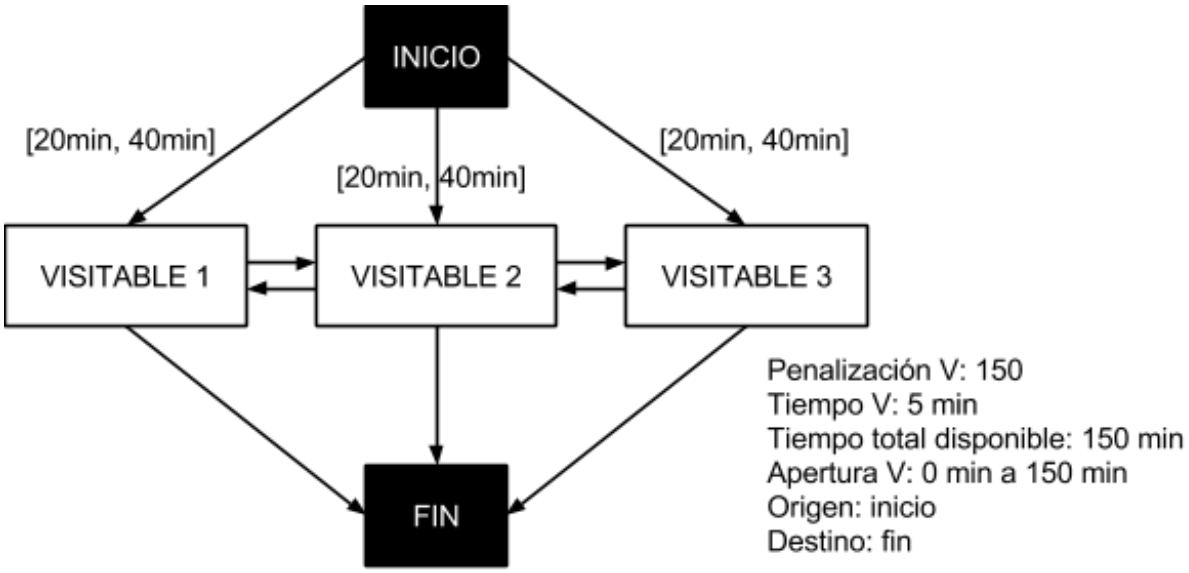

Figura 3.13: Ejemplo - ocupación temporal del plan

```
0.001: (move origin p visitable3)  [5.000]
5.002: (visit visitable3 p)  [20.000]
25.003: (move visitable3 p visitable2)  [5.000]
30.004: (visit visitable2 p)  [20.000]
50.005: (move visitable2 p visitable1)  [5.000]
55.006: (visit visitable1 p)  [20.000]
75.007: (move visitable1 p finish)  [5.000]
```

Por el contrario, si se busca maximizar la ocupación temporal las visitas deberán tener un valor cercano a 40 min, el plan devuelto por el recomendador así lo hace como se muestra a continuación:

```
; Plan foung with metric 14500.000
; States evaluated so far: 1295
; States pruned bases on preheuristic cost lower bound: 0
; Time 7.55
0.001: (move origin p visitable3)  [5.000]
5.002: (visit visitable3 p)  [40.000]
45.003: (move visitable3 p visitable2)  [5.000]
50.004: (visit visitable2 p)  [40.000]
90.005: (move visitable2 p visitable1)  [5.000]
95.006: (visit visitable1 p)  [39.999]
135.006: (move visitable1 p finish)  [5.000]
```

Concluyendo se ha observado que todas las penalizaciones funcionan correctamente por separado, las pruebas finales usando el global de penalizaciones a la vez se encuentran en el capítulo de pruebas.

Capítulo 4

Experimentos

En este capítulo, se describen los experimentos realizados para validar el enfoque planteado en este trabajo. Teniendo en cuenta que la principal novedad de este trabajo radica en la implementación de un método para tener en cuenta las preferencias del usuario con respecto a la configuración de una agenda turística, el objetivo principal de esta evaluación es el análisis del comportamiento del módulo de planificación, es decir, la adaptación de los planes obtenidos a las preferencias del usuario en cuanto al número de visitas a incluir y el nivel de ocupación temporal.

Concretamente, se ha escogido cuatro usuarios distintos, para los que se ha obtenido la recomendación de actividades e intervalos de duración con GRSK. Dicha recomendación se muestra en la tabla 4.1. Además, cada usuario presenta diferentes requerimientos en cuanto a tiempo disponible, tiempo para comer, medio de transporte, y lugares de origen y destino, tal y como se muestra en la segunda columna de la tabla 4.2. Para cada usuario, se ha generado un plan con todas las posibles combinaciones en cuanto a preferencias sobre número de visitas y nivel de ocupación, es decir, 9 configuraciones distintas. Un resumen de los resultados obtenidos se muestra en la tabla 4.2, donde:

- La columna **#Visits** indica la preferencia dada por el usuario con respecto al número de visitas a incluir en el plan, que puede contener tres valores que expresan una preferencia por realizar muchas visitas (*Many*), una preferencia por realizar pocas visitas (*A few*) o le es indiferente (*Indifferent*).

- La columna **Occupation** indica la preferencia dada por el usuario con respecto al nivel de ocupación temporal del plan, es decir, si prefiere obtener planes en los que se utiliza la mayor parte del tiempo disponible (*High*), planes en los que se dispone de más tiempo libre (*Low*) o le es indiferente (*Indifferent*).

- La columna **Move (%)** indica, dado un plan, el porcentaje del tiempo de dicho plan que se dedica a acciones de desplazamiento. Como se ha comentado anteriormente, el planificador busca reducir este valor lo máximo posible.

- La columna **Visits (%)** indica, dado un plan, el porcentaje del tiempo de dicho plan

	User 1	User 2	User 3	User 4
Museo de las Ciencias Príncipe Felipe	X			X
El Oceanográfico	X	X		X
El Almudín	X		X	
La Catedral	X	X		X
Jardín Botánico	X		X	
Mercado Central	X		X	
Museo de Bellas Artes de Valencia-San Pio V	X	X	X	X
Centro Julio Gonzalez		X	X	
La Lonja		X	X	X
Torre del Miguelete		X	X	
Museo de Cerámica (Palacio Marqués de Dos Aguas)		X		

Cuadro 4.1: Lugares recomendados para cada usuario

empleado en visitar los lugares recomendados. En este caso, este valor se verá afectado por la preferencia particular del usuario.

- La columna **Occupation (%)** es el ratio entre la ocupación total en el plan y el tiempo total disponible. Es decir, en el valor de la ocupación total se considera la suma del tiempo empleado en la realización de las visitas, tiempo empleado en el desplazamiento entre las localizaciones y el tiempo empleado en la comida.
- La columna **#Places** indica el número de lugares visitados en el plan obtenido.
- La columna **Utility (%)** muestra el ratio entre la suma de las preferencias de los lugares visitados en dicho plan con respecto a la suma de las preferencias de todos los lugares recomendados.

En primer lugar, analizaremos con más detalle varios casos de estudio con el usuario 1, es decir, estudiaremos la adaptación del plan que realiza OPTIC de acuerdo con las distintas preferencias. El sistema de recomendación GRSK ha recomendado el conjunto de actividades que se muestra en la tabla 4.1, junto con su grado de interés estimado (o prioridad) y el intervalo de duración recomendado. Los cuatro casos de estudio seleccionados corresponden a los casos C1, C3, C7 y C38 en la tabla 4.2.

El primer caso de estudio C1 (Figura 4.1) es el caso donde el usuario no especifica ninguna preferencia sobre la configuración de la agenda, por lo tanto el planificador se centra en buscar una solución en la que se reduzca el tiempo de desplazamiento y que incluya los lugares recomendados que maximicen la utilidad. El resultado muestra un tiempo de desplazamiento del 15 %, una ocupación temporal del 90,625 % y una utilidad del 63,60424 %. Estos resultados demuestran que, efectivamente, se ha conseguido un tiempo de desplazamiento muy bajo y una ocupación alta, lo que permite obtener una utilidad difícil de superar, como se mostrará más adelante.

El segundo caso de estudio C2 (Figura 4.2) se corresponde con el caso en el que el usuario especifica que desea una alta ocupación temporal. Esto se traduce en un aumento de la ocupación temporal comparado con el caso C1, hasta llegar al 99,9875 %. El tiempo dedicado a las visitas también se ha incrementado hasta el 72,69583 %. Analizando las

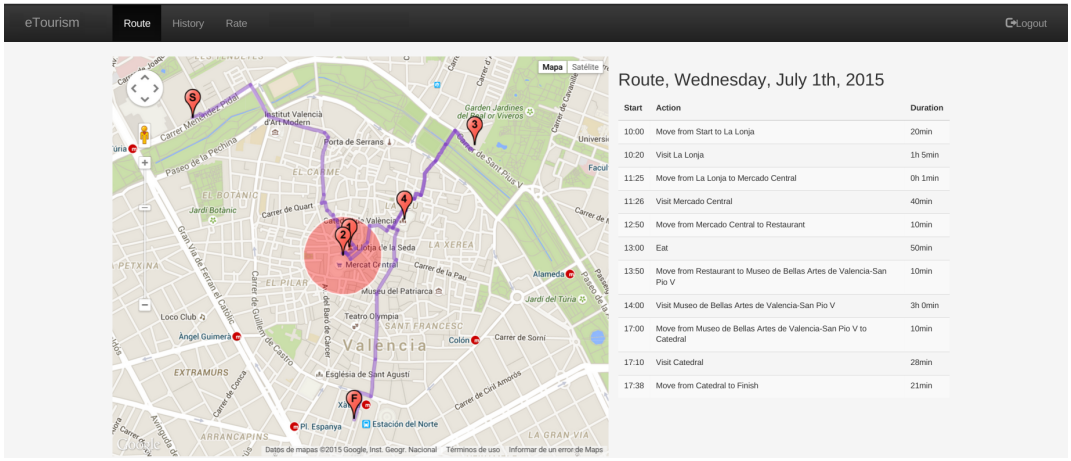

Figura 4.1: Plan generado para el caso 1

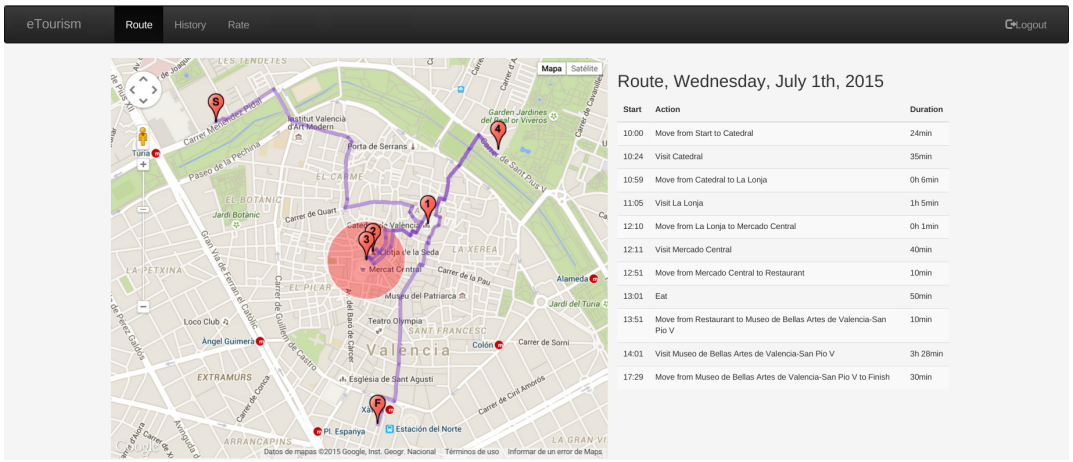

Figura 4.2: Plan generado para el caso 2

figuras 4.1 y 4.2, se observa que la principal diferencia entre ambos planes (aparte del orden de las visitas) radica en el incremento en el tiempo de visita de dos lugares (*Catedral* y *Museo de Bellas Artes*).

En el tercer caso de estudio C7 (Figura 4.3), se parte del caso inicial, con la diferencia de que el usuario especifica que desea visitar pocos lugares. Esto se ha traducido en una reducción del número de lugares de 4 a 3, que deriva en una reducción del tiempo de desplazamiento 8,958333 %, ya que se incluirá una acción de desplazamiento menos en el plan, y en una reducción en el tiempo de visita hasta el 23,333334 %, ya que se incluirá una acción de visita menos en el plan. Como consecuencia, la utilidad también se ha visto afectada, y se ha reducido hasta alcanzar el 49,469967 %.

Finalmente, se estudia el caso C37 de la tabla 4.2 (Figura 4.4) que parte del caso base C1, donde se ha realizado una modificación en el modo de desplazamiento: en el caso C1 el

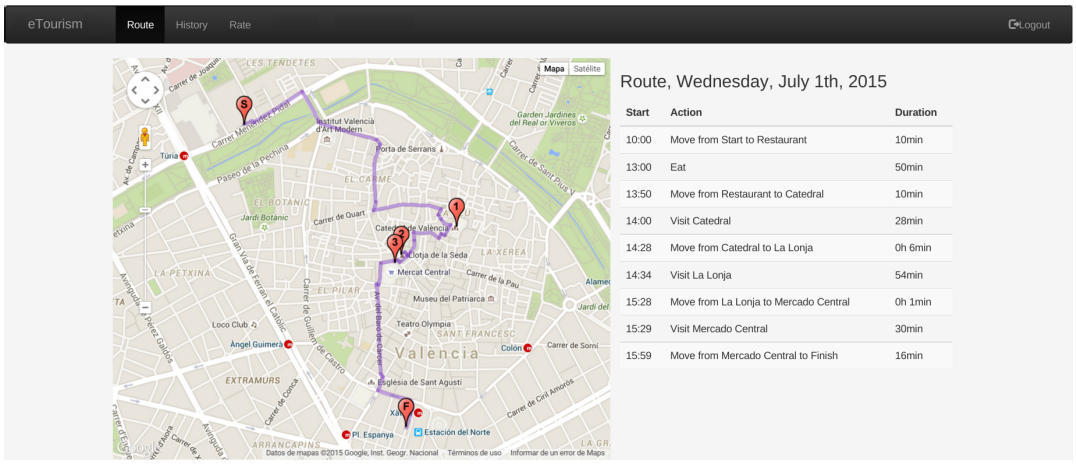

Figura 4.3: Plan generado para el caso 7

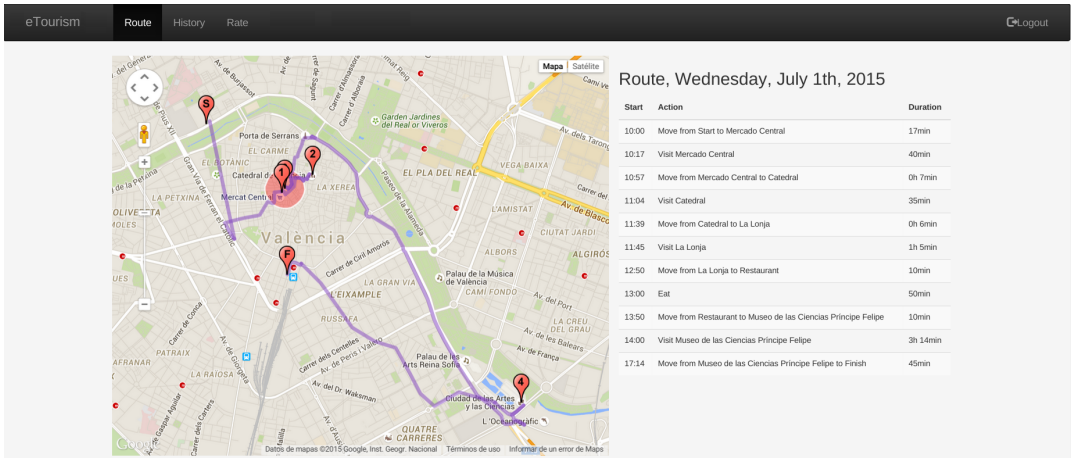

Figura 4.4: Plan generado para el caso 38

usuario se desplazaba andando y en el caso C37 utiliza el modo mixto, es decir, andando cuando la distancia sea menor de un umbral y en transporte público en otro caso. La ventaja de este modo mixto es que permite una reducción en el tiempo de desplazamiento (del 15 % hasta el 10,208333 % en este caso) y, además, permite visitar lugares más lejanos (en este caso el *Museo de las Ciencias Principe Felipe*), que de otro modo, sería imposible visitar dada la restricción del tiempo disponible.

Desde un punto de vista más global, como se puede observar en la tabla 4.2, en todos los casos los tiempos de desplazamiento son inferiores al 22 % del tiempo total disponible exceptuando los resultados obtenidos para el usuario 2. En este caso, el porcentaje del tiempo de desplazamiento aumenta hasta el 47 %. Esto se debe fundamentalmente a que, en primer lugar, el origen y destino del plan se ubicaron significativamente más lejos que en los demás casos. En segundo lugar, el medio de transporte escogido en este caso es el

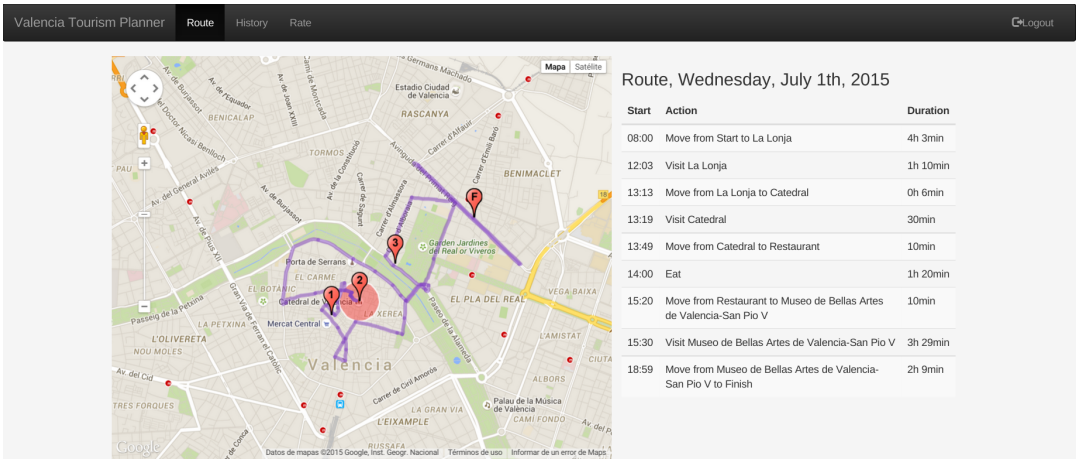

Figura 4.5: Plan generado para el caso 14

coche y, dado que las rutas generadas para el usuario 2 solo contenían visitas dentro del casco histórico, esto ha resultado en rutas altamente ineficientes, tal y como se muestra en el ejemplo del caso C14 en la figura 4.5.

El tiempo empleado en las visitas a los lugares turísticos llega a alcanzar el 73 % del tiempo total disponible, que se considera un tiempo alto teniendo en cuenta que cada lugar tiene unos tiempos mínimos y máximos de visita que no pueden ser excedidos y por lo tanto es casi imposible conseguir una combinación que llegue a alcanzar un 100 %. Además, se debe contar con que a ese 100 % se le debe restar el tiempo de desplazamiento y el tiempo dedicado a la comida. El usuario 2 es el único que muestra una reducción considerable en el tiempo de visita y es debido al alto porcentaje de tiempo invertido en el desplazamiento.

La ocupación total del tiempo del plan está en la mayoría de los casos por encima del 85 % de tiempo total, alcanzando en algunos casos el 99 %. Estos casos con una ocupación tan alta, obviamente coinciden con los planes que incluyen un mayor número de visitas (4 visitas). Los casos con menor número de visitas coinciden acertadamente con configuraciones donde la preferencia del usuario con respecto al número de visitas es *A few*. La utilidad final del plan, obtenida como el porcentaje de la suma de las prioridades de los lugares incluidos en el plan con respecto al total de la prioridad de los lugares recomendados, oscila entre el 40 % y el 67 %. Hay que tener en cuenta de nuevo, que existe una cota superior en la utilidad a alcanzar, ya que el plan no solamente incluye visitas, sino también acciones de desplazamiento y la acción de comer. Por ello, se ha realizado un estudio más detallado de la *utilidad por minuto* de un plan. Así, esta utilidad, para un plan calculado, se obtiene como:

$$U^m_{plan} = \frac{\sum_{\forall a \in \Pi} Pr^a}{T_{initial}}$$

Por otro lado, debemos obtener la *utilidad por minuto teórica* para un problema dado.

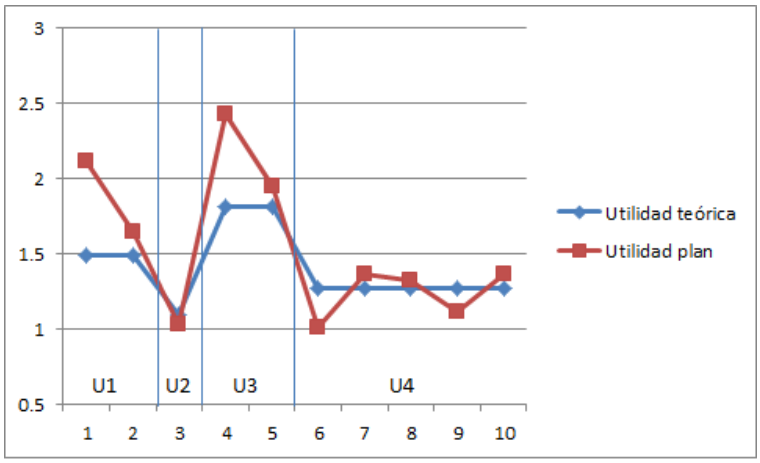

Figura 4.6: Comparativa entre la utilidad teórica y la utilidad real de los planes obtenidos

Así, dado el conjunto de visitables RA, se calcula la duración de cada visita como el promedio del intervalo recomendado y se define TV como el tiempo total necesario para realizar todas las visitas. El tiempo necesario para obtener un 100 % de utilidad sería:

$$TT = TV + TV * \alpha + TC$$

Donde α es el porcentaje del plan real empleado en desplazamiento y TC es el tiempo empleado en la comida. Por tanto, la *utilidad por minuto teórica* será:

$$U^m_{teorica} = \frac{\sum_{\forall a \in RA} Pr^a}{TT}$$

La figura 4.6 muestra la comparativa entre la utilidad por minuto téorica y real para cada uno de los planes distintos obtenidos para cada usuario. Como se puede observar, la utilidad real del plan supera en algunos casos a la utilidad teórica. Esto es debido a que en el plan real se han incluido los lugares con mayor prioridad, lo que es el comportamiento deseable para la aplicación. En los casos en los que la utilidad real queda por debajo de la utilidad teórica, esta diferencia no es especialmente remarcable.

Sin embargo, se observan algunos casos puntuales donde los planes obtenidos no son completamente acordes a las preferencias del usuario. En general, esto se observa en la combinación de preferencias *A few/Low*, donde se obtienen valores de #visitas y de % ocupación similares a otras opciones. Esto se debe a que la penalización de la utilidad domina sobre la combinación de las penalizaciones por preferencias. Es decir, cuando solamente entra en juego una de las penalizaciones por preferencias, el resultado obtenido sí es acorde a dicha penalización, pero es la combinación de ambas con valores *A few/Low* la que impide el cálculo de un plan perfectamente adaptado a dichas preferencias.

Por otro lado, para el usuario 4, se observa cómo la preferencia sobre el número de visitas se tiene en cuenta de forma adecuada. Los planes en los que se solicita un número

de visitas alto se obtienen planes con 4 lugares, mientras que en los se prefiere un número de visitas bajo, los planes incluyen únicamente 2 o 3 lugares.

En definitiva, se observa cómo OPTIC no es capaz de manejar de forma completamente correcta una combinación de preferencias. Sí resuelve de manera eficiente el problema de planificación temporal que se deriva de la construcción de la agenda turística, y es capaz de tener en cuenta los intervalos de duración y los horarios de apertura y cierre de los lugares. También maneja correctamente el intervalo de parada a comer y las actividades de movimiento. En cuanto a preferencias, las prioridades de los lugares recomendados son el elemento fundamental en la selección, tal y como demuestra el porcentaje de utilidad obtenido. El orden entre las visitas viene claramente determinado por la distancia entre ellas, gracias a la penalización por desplazamiento. Sin embargo, la combinación de otras preferencias a un nivel más bajo (nivel de ocupación y número de visitas) no se resuelve de forma completamente correcta. Así:

- Se observa cierta dificultad para modelizar adecuadamente la ocupación temporal, esto es, los huecos libres entre actividades. Debido a una limitación de OPTIC, se consideran todas las actividades, incluyendo el transporte y comidas (la especificación se hace como `tiempo_total/tiempo_disponible`). Esto no garantiza los *huecos* entre actividades sino que OPTIC tenderá a dar un valor más alto o más bajo a `tiempo_total` y esto lo consigue poniendo más lugares a visitar (valores altos `tiempo_total`) o poniendo menos lugares a visitar (valores bajos de `tiempo_total`). Por tanto, el peso de esta preferencia está estrechamente relacionado con el de número de lugares a visitar.

- En líneas generales, cuando el usuario selecciona *Indifferent*, el planificador tiene menos preferencias que satisfacer, en particular las variables `tiempo_total` y *número de lugares visitados* no influyen, por lo que el conjunto de preferencias se limitan a las visitas y el tiempo de transporte. Siendo las preferencias sobre las visitas las que tienen mayor peso en la función objetivo, OPTIC tenderá a introducir el mayor número de lugares a visitar para minimizar la función objetivo. Sin embargo, se puede observar que las opciones *Many/High* reflejan un mayor valor de %visit y %occupation, lo que es consistente con las preferencias/gustos del usuario.

En resumen, cuanto menos constreñido esté el planificador en el conjunto de preferencias a satisfacer, OPTIC tiende a satisfacer el resto de preferencias, que son fundamentalmente las recomendaciones del SR. Esto conduce, en líneas generales, a planes que suelen contener un elevado número de lugares a visitar. Sin embargo, si lo comparamos con las opciones *Many/High*, se puede observar que, efectivamente, estas opciones tienen una influencia en los valores %visit y %occupation respecto a las opciones *Indifferent/Indifferent*, ya que se consiguen valores más altos de %visit y %occupation. Esto indica que, efectivamente, las preferencias del usuario respecto al número de lugares a visitar y tiempo de ocupación tienen un peso en la función objetivo.

		Preferences		Results				
C	**Users**	**#Visits**	**Occupation**	**Move(%)**	**Visit(%)**	**Occupation(%)**	**Places**	**Utility (%)**
1		Indifferent	Indifferent	15,000001	65,08336	90,625	4	63,60424
2		Indifferent	High	16,875	72,69583	99,9875	4	63,60424
3	**User 1**	Indifferent	Low	16,25	72,815414	99,482086	4	63,60424
4	**Start: 10:00**	Many	Indifferent	15,000001	65,208336	90,625	4	63,60424
5	**Finish: 18:00**	Many	High	16,875	72,69583	99,9875	4	63,60424
6	**Eat: 1 hour**	Many	Low	16,25	72,815414	99,482086	4	63,60424
7	**Walk**	A few	Indifferent	8,958333	23,333334	42,708336	3	49,469967
8		A few	High	16,875	72,69583	99,9875	4	63,60424
9		A few	Low	16,25	72,815414	99,482086	4	63,60424
10		Indifferent	Indifferent	47,38095	33,451668	90,35643	3	42,774567
11		Indifferent	High	47,38095	37,012733	93,917496	3	42,774567
12	**User 2**	Indifferent	Low	47,38095	36,92226	93,82703	3	42,774567
13	**Start: 08:00**	Many	Indifferent	47,38095	33,451668	90,35643	3	42,774567
14	**Finish: 22:00**	Many	High	47,38095	37,012733	93,917496	3	42,774567
15	**Eat: 1,5 hours**	Many	Low	47,38095	36,92226	93,82703	3	42,774567
16	**Driving**	A few	Indifferent	47,38095	33,451668	90,35643	3	42,774567
17		A few	High	47,38095	37,012733	93,917496	3	42,774567
18		A few	Low	47,38095	36,92226	93,82703	3	42,774567
19		Indifferent	Indifferent	21,666666	57,617855	98,332146	4	56,98324
20		Indifferent	High	21,666666	59,272854	99,98714	4	56,98324
21	**User 3**	Indifferent	Low	21,666666	59,182613	99,896904	4	56,98324
22	**Start: 10:00**	Many	Indifferent	21,666666	57,617855	98,332146	4	56,98324
23	**Finish: 17:00**	Many	High	21,666666	59,272854	99,98714	4	56,98324
24	**Eat: 1,5 hour**	Many	Low	21,666666	59,182613	99,896904	4	56,98324
25	**Public transport**	A few	Indifferent	20,47619	44,76071	84,28452	3	45,810055
26		A few	High	21,666666	59,272854	99,98714	4	56,98324
27		A few	Low	21,666666	59,182613	99,896904	4	56,98324
28		Indifferent	Indifferent	6,666667	46,11111	63,88889	4	67,84452
29		Indifferent	High	12,777779	62,071945	85,96083	4	67,84452
30	**User 4**	Indifferent	Low	9,722222	62,398613	83,23194	4	67,84452
31	**Start: 10:00**	Many	Indifferent	12,777779	62,071945	85,96083	4	67,84452
32	**Finish: 22:00**	Many	High	12,777779	62,071945	85,96083	4	67,84452
33	**Eat: 1 hour**	Many	Low	9,722222	62,398613	83,23194	4	67,84452
34	**Cycling**	A few	Indifferent	3,7500002	55,555557	70,416664	2	36,39576
35		A few	High	9,444445	65,40569	85,96124	3	52,650177
36		A few	Low	9,444445	65,315414	85,87097	3	52,650177
37		Indifferent	Indifferent	10,208333	44,583332	65,208336	4	55,172413
38		Indifferent	High	12,083333	42,071457	64,57146	3	40,75235
39	**User 1**	Indifferent	Low	12,083333	41,98104	64,48104	3	40,75235
40	**Start: 10:00**	Many	Indifferent	8,541667	30,833334	49,791668	3	42,006268
41	**Finish: 18:00**	Many	High	12,083333	42,071457	64,57146	3	40,75235
42	**Eat: 1 hour**	Many	Low	12,083333	41,98104	64,48104	3	40,75235
43	**Mix mode**	A few	Indifferent	7,9166665	17,291666	35625	2	29,467085
44		A few	High	9,375	44,78021	64,57188	2	26,959246
45		A few	Low	9375	44,689377	64,48104	2	26,959246

Cuadro 4.2: Result Table

Capítulo 5

Conclusión

5.1. Introducción

El siguiente documento ha detallado el proceso y los componentes que se han creado para la aplicación Valencia Tourism Planner. Los objetivos inicialmente planteados se han cubierto satisfactoriamente.

Así, se ha creado una aplicación capaz de generar rutas personalizadas para turistas en la ciudad de Valencia, teniendo en cuenta tanto sus preferencias a nivel del tipo de lugares a visitar como sus preferencias a nivel de la configuración de la agenda. Concretamente, se han realizado las siguientes tareas:

- Se ha implementado un mecanismo de recuperación de la información de la ciudad de Valencia mediante los servicios que proporciona Google. Por ejemplo, horarios de apertura y cierre de los lugares, cálculo de una ruta óptima entre dos lugares, etc.

- Se ha diseñado una nueva interfaz para la introducción de las preferencias por parte del usuario, así como para mostrar la agenda calculada de acuerdo con estas preferencias. La agenda se muestra como una lista de lugares a visitar, junto con la hora de comienzo de la visita y la duración aconsejada. Los desplazamientos se muestran en un mapa indicando las calles por las que el usuario debe desplazarse.

- Se ha especificado de una tarea de planificación a partir de los datos de entrada personalizados del usuario y de sus preferencias sobre la ruta. Esto supone la generación automática del dominio y problema de planificación en lenguaje PDDL a partir de los datos del usuario y la información recuperada de la ciudad de Valencia.

Desde el punto de vista personal, la realización de este proyecto me ha proporcionado una mayor formación y práctica en el campo de los sistemas de recomendación y planificación automática, en los que tengo interés en seguir trabajando.

5.2. Avances y ventajas

VTP se puede considerar una gran ayuda para el turismo en la ciudad de Valencia ofreciendo los siguientes avances:

- La aplicación ofrece una recomendación de una agenda completamente personalizada a los gustos y preferencias del usuario.
- La aplicación es independiente, no requiere que se modifiquen manualmente los elementos de la base de datos puesto que lo hace de forma automática, si lo necesita.
- VTP es un sistema que combina dos campos de la Inteligencia Artificial, sistemas de recomendación y planificación automática que permite obtener rutas que satisfacen las preferencias del usuario desde dos puntos de vista distintos: gustos personales en las visitas y preferencias en la configuración de la agenda.

5.3. Trabajo futuro y mejoras

VTP es una aplicación en proceso de desarrollo sobre la que se va a seguir trabajando y añadiendo las siguientes mejoras:

- Internacionalización de la aplicación: este prototipo utiliza sólo el idioma inglés; puesto que recomienda rutas por Valencia, se debería adaptar al menos al valenciano y al castellano.
- Extender VTP para que pueda utilizar otros planificadores: la forma de trabajar de OPTIC requiere un gran tiempo para generar un buen plan. En el grupo de investigación con el que se ha colaborado para realizar este trabajo, se está desarrollando un nuevo planificador temporal basado en FLAP [26]. Éste se podría modificar para que aceptase preferencias y probar su funcionamiento bajo este dominio.
- Modelar este problema utilizando un CSP, una vez comprobado cómo funciona esta aplicación bajo un planificador temporal con preferencias, y comparar resultados obtenidos con ambas técnicas.
- Incluir Ajax, ya que esta tecnología permitiría al sistema ir mostrando las rutas al usuario a medida que se van obteniendo (el planificador genera varios planes, incrementando la calidad de los mismos) y éste podría cancelar el proceso de planificación cuando se le mostrara una ruta que satisfaciera sus preferencias.
- Incluir tecnología de detección de fenómenos externos, por ejemplo detección de atascos, detección de condiciones climáticas extremas y actuar en consecuencia.
- Dado que la base de la aplicación está ya creada, se podría adaptar para que funcionase en distintas ciudades.

Capítulo 6

Anexos

6.1. Anexo 1: dominio PDDL

A continuación se muestra el dominio empleado para la implementación del Planificador Inteligente que permite resolver el problema de esta tesis final de master.

```
(define (domain tourist)
  (:requirements
    :typing
    :durative-actions
    :fluents
    :preferences
    :constraints
    :timed-initial-literals
    :duration-inequalities
    :equality
  )

  (:types
    person
    visitable restaurant hotel - location
  )

  (:predicates
    (be ?x - person  ?y - location)
    (visit_location ?x - location)
    (not_visit_location ?x - location)
    (eaten)
    (not_eaten)
    (open ?x - location)
    (active)
(can_move)
  )

  (:functions
    (location_time ?x - location ?y - location)
    (min_visit_time ?y - location)
    (max_visit_time ?y - location)
```

```
    (eat_time)
    (total_available_time)
    (number_visit_location)
    (transport_time)
  )

  (:durative-action move
    :parameters (?x - location ?y - person ?z - location)
    :duration (= ?duration (location_time ?x ?z))
    :condition
      (and
        (over all (active))
        (at start (can_move))
        (at start (be ?y ?x))
        (at start (>= (total_available_time) (location_time ?x ?z)))
      )
    :effect
      (and
        (at start (not (be ?y ?x)))
        (at end (be ?y ?z))
        (at end (decrease (total_available_time) (location_time ?x ?z)))
        (at end (increase (transport_time) (location_time ?x ?z)))
        (at end (not (can_move)))
      )
  )

  (:durative-action visit
    :parameters (?x - location ?y - person)
    :duration
      (and
        (>= ?duration (min_visit_time ?x))
        (<= ?duration (max_visit_time ?x))
        (<= ?duration (total_available_time))
      )
    :condition
      (and
        (over all (active))
        (at start (not_visit_location ?x))
        (over all (be ?y ?x))
        (over all (open ?x))
      )
    :effect
      (and
        (at end (can_move))
        (at start (not (not_visit_location ?x)))
        (at end (visit_location ?x))
        (at end (decrease (total_available_time) ?duration))
        (at end (increase (number_visit_location) 1))
      )
  )

  (:durative-action eat
    :parameters (?x - restaurant ?y - person)
    :duration (= ?duration(eat_time))
```

```
    :condition
      (and
        (over all (not_eaten ?y))
        (over all (active))
        (over all (be ?y ?x))
        (over all (open ?x))
        (at start (>= (total_available_time) (eat_time)))
      )
    :effect
      (and
        (at end (can_move))
        (at end (not (not_eaten ?y)))
        (at end (eaten ?y))
        (at end (decrease (total_available_time) (eat_time))
      )
    )
  )
)
```

6.2. Anexo 2: problema PDDL

En cuanto a los problemas cada uno de ellos es distinto y es generado automáticamente por la aplicación web, a continuación se muestra un ejemplo de uno de ellos:

```
(define (problem problem1)

(:domain tourist)

(:objects
  P - person
  id_17380 - visitable
  id_17374 - visitable
  id_17396 - visitable
  id_17419 - visitable
  id_17431 - visitable
  id_17402 - visitable
  id_17375 - visitable
  id_55 - restaurant
  id_01 id_99 - hotel
)

(:init
  (be P id_01)

  (at 0  (open id_17380))
  (at 1080 (not (open id_17380)))

  (at 0  (open id_17374))
  (at 1080 (not (open id_17374)))

  (at 0  (open id_17396))
  (at 420 (not (open id_17396)))
```

```
(at 0  (open id_17419))
(at 1080 (not (open id_17419)))

(at 0  (open id_17431))
(at 1080 (not (open id_17431)))

(at 0  (open id_17402))
(at 1080 (not (open id_17402)))

(at 0  (open id_17375))
(at 1080 (not (open id_17375)))

(at 180 (open id_55))
(at 240 (not (open id_55)))

(= (location_time id_01 id_17380) 86)
(= (location_time id_01 id_17374) 92)
(= (location_time id_01 id_17396) 24)
(= (location_time id_01 id_17419) 24)
(= (location_time id_01 id_17431) 10)
(= (location_time id_01 id_17402) 20)
(= (location_time id_01 id_17375) 29)

(= (location_time id_99 id_17380) 54)
(= (location_time id_99 id_17374) 60)
(= (location_time id_99 id_17396) 22)
(= (location_time id_99 id_17419) 21)
(= (location_time id_99 id_17431) 25)
(= (location_time id_99 id_17402) 17)
(= (location_time id_99 id_17375) 30)

(= (location_time id_55 id_17380) 10)
(= (location_time id_55 id_17374) 10)
(= (location_time id_55 id_17396) 10)
(= (location_time id_55 id_17419) 10)
(= (location_time id_55 id_17431) 10)
(= (location_time id_55 id_17402) 10)
(= (location_time id_55 id_17375) 10)
(= (location_time id_17380 id_55) 10)
(= (location_time id_17374 id_55) 10)
(= (location_time id_17396 id_55) 10)
(= (location_time id_17419 id_55) 10)
(= (location_time id_17431 id_55) 10)
(= (location_time id_17402 id_55) 10)
(= (location_time id_17375 id_55) 10)
(= (location_time id_55 id_01) 10)
(= (location_time id_01 id_55) 10)
(= (location_time id_55 id_99) 10)
(= (location_time id_99 id_55) 10)

(= (location_time id_17380 id_17374) 6)
(= (location_time id_17380 id_17396) 63)
(= (location_time id_17380 id_17419) 63)
```

```
(= (location_time id_17380 id_17431) 78)
(= (location_time id_17380 id_17402) 65)
(= (location_time id_17380 id_17375) 65)

(= (location_time id_17374 id_17380) 6)
(= (location_time id_17374 id_17396) 69)
(= (location_time id_17374 id_17419) 69)
(= (location_time id_17374 id_17431) 84)
(= (location_time id_17374 id_17402) 71)
(= (location_time id_17374 id_17375) 71)

(= (location_time id_17396 id_17380) 63)
(= (location_time id_17396 id_17374) 69)
(= (location_time id_17396 id_17419) 2)
(= (location_time id_17396 id_17431) 21)
(= (location_time id_17396 id_17402) 9)

(= (location_time id_17396 id_17375) 8)
(= (location_time id_17419 id_17380) 63)
(= (location_time id_17419 id_17374) 69)
(= (location_time id_17419 id_17396) 2)
(= (location_time id_17419 id_17431) 19)
(= (location_time id_17419 id_17402) 7)
(= (location_time id_17419 id_17375) 10)

(= (location_time id_17431 id_17380) 78)
(= (location_time id_17431 id_17374) 84)
(= (location_time id_17431 id_17396) 21)
(= (location_time id_17431 id_17419) 19)
(= (location_time id_17431 id_17402) 13)
(= (location_time id_17431 id_17375) 27)

(= (location_time id_17402 id_17380) 65)
(= (location_time id_17402 id_17374) 71)
(= (location_time id_17402 id_17396) 9)
(= (location_time id_17402 id_17419) 7)
(= (location_time id_17402 id_17431) 13)
(= (location_time id_17402 id_17375) 17)

(= (location_time id_17375 id_17380) 65)
(= (location_time id_17375 id_17374) 71)
(= (location_time id_17375 id_17396) 8)
(= (location_time id_17375 id_17419) 10)
(= (location_time id_17375 id_17431) 27)
(= (location_time id_17375 id_17402) 17)

(= (location_time id_17380 id_01) 86)
(= (location_time id_17374 id_01) 92)
(= (location_time id_17396 id_01) 24)
(= (location_time id_17419 id_01) 24)
(= (location_time id_17431 id_01) 10)
(= (location_time id_17402 id_01) 20)
(= (location_time id_17375 id_01) 29)
```

```
  (= (location_time id_17380 id_99) 54)
  (= (location_time id_17374 id_99) 60)
  (= (location_time id_17396 id_99) 22)
  (= (location_time id_17419 id_99) 21)
  (= (location_time id_17431 id_99) 25)
  (= (location_time id_17402 id_99) 17)
  (= (location_time id_17375 id_99) 30)

  (= (min_visit_time id_17380) 170)
  (= (min_visit_time id_17374) 230)
  (= (min_visit_time id_17396) 55)
  (= (min_visit_time id_17419) 28)
  (= (min_visit_time id_17431) 55)
  (= (min_visit_time id_17402) 65)
  (= (min_visit_time id_17375) 180)

  (= (max_visit_time id_17380) 220)
  (= (max_visit_time id_17374) 290)
  (= (max_visit_time id_17396) 70)
  (= (max_visit_time id_17419) 35)
  (= (max_visit_time id_17431) 66)
  (= (max_visit_time id_17402) 75)
  (= (max_visit_time id_17375) 210)

  (not_visit_location id_17380)
  (not_visit_location id_17374)
  (not_visit_location id_17396)
  (not_visit_location id_17419)
  (not_visit_location id_17431)
  (not_visit_location id_17402)
  (not_visit_location id_17375)

  (at 0 (active))
  (at 480 (not (active)))

  (= (eat_time) 50)
  (= (total_available_time) 480)

  (= (number_visit_location) 0)

  (= (transport_time) 0)

  (can_move)

  (not_eaten P)

)

(:goal
  (and
    (be P id_99)
    (eaten P)

    (preference p1 (visit_location id_17380))
```

```
    (preference p2 (visit_location id_17374))
    (preference p3 (visit_location id_17396))
    (preference p4 (visit_location id_17419))
    (preference p5 (visit_location id_17431))
    (preference p6 (visit_location id_17402))
    (preference p7 (visit_location id_17375))

    (> (number_visit_location) 0)

 )
)

(:metric minimize
  (+
    (* (/ (* 37 (is-violated p1)) 7) 100)
    (* (/ (* 36 (is-violated p2)) 7) 100)
    (* (/ (* 34 (is-violated p3)) 7) 100)
    (* (/ (* 32 (is-violated p4)) 7) 100)
    (* (/ (* 30 (is-violated p5)) 7) 100)
    (* (/ (* 28 (is-violated p6)) 7) 100)
    (* (/ (* 28 (is-violated p7)) 7) 100)

    (* (/ (transport_time) 480) 100)

    (* 0 (* (/ (number_visit_location) 7) 100))
    (* 0 (* (/ (total_available_time) 480) 100))

 )
)
)
```

Bibliografía

[1] E. Onaindia A. Garrido. On the application of least-commitment and heuristic search in temporal planning. In *International Joint Conference on AI (IJCAI-2003)*, pages 942–947. Morgan Kaufmann, 2003.

[2] D. Isern A. Moreno, A. Valls, L. Marin, and J. Borr'as. Sigtur/e-destination: ontology-based personalized recommendation of tourism and leisure activities. *Engineering Applications of Artificial Intelligence*, 26, 2013.

[3] Javier Bejar, Sergio Alvarez, Dario Garcia-Gasulla, Ignasi Gomez, Luis Oliva-Felipe, and Arturo Tejeda. Discovery of spatio-temporal patterns from location based social networks. In *17th International Conference of the Catalan Association for Artificial Intelligence*, volume Frontiers in Artificial Intelligence and Applications, Volume 269: Artificial Intelligence Research and Development, pages 126–135, 2014.

[4] J. Benton, Amanda Jane Coles, and Andrew Coles. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*, 2012.

[5] B. Bonet and H. Geffner. Heuristics for planning with penalties and rewards using compiled knowledge. *10th Int. Conf. on Knowledge Representation (KR'06)*, 2006.

[6] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[7] Luis A. Castillo, Eva Armengol, Eva Onaindia, Laura Sebastia, Jesús González-Boticario, Antonio Rodríguez, Susana Fernández, Juan D. Arias, and Daniel Borrajo. samap: An user-oriented adaptive system for planning tourist visits. *Expert Syst. Appl.*, 34(2):1318–1332, 2008.

[8] Amanda Jane Coles, Andrew Coles, Maria Fox, and Derek Long. Forward-chaining partial-order planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, pages 42–49, 2010.

[9] Stefan Edelkamp and Jorg Hoffmann. Pddl2.2: The language for the classical part of ipc-4. 2004.

[10] D. McDermott *et al.* PDDL - the planning domain definition language. Technical report, Yale University, 1998.

[11] Josef Fink and Alfred Kobsa. A review and analysis of commercial user modeling servers for personalization on the world wide web. *User Modeling and User-Adapted Interaction*, 10(2-3):209–249, 2000.

[12] Maria Fox and Derek Long. Pddl+ : Modelling continuous time-dependent effects.

[13] Maria Fox and Derek Long. pddl2.1 : An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20, 2003.

[14] Adomavicius G. and Tuzhilin A. Personalization technologies: a process-oriented perspective. *Communications of the ACM*, 48, 2005.

[15] Inma Garcia, Laura Sebastia, and Eva Onaindia. On the design of individual and group recommender systems for tourism. *Expert Systems with Applications*, 38(6):7683–7692, 2011.

[16] Alfonso Gerevini, Patrik Haslum, Derek Long, Alessandro Saetti, and Yannis Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.*, 173(5-6):619–668, 2009.

[17] E. Giunchiglia and M. Maratea. Planning as satisfiability with preferences. *22nd Conf. of Artificial Intelligence (AAAI'07)*, 2007.

[18] Oki B.M. Goldberg D., Nichols D. and Ferry D. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35, 1992.

[19] Google Inc. The google directions api, https://developers.google.com/maps/documentation/directions/.

[20] Google Inc. The google maps api, https://developers.google.com/maps/.

[21] Google Inc. The google places api, https://developers.google.com/places/webservice/search.

[22] A. Moreno J. Borr'as and A. Valls. Intelligent tourism recommender systems: A survey. *Expert Syst*, 41.

[23] Yohei Kurata. Ct-planner2: More flexible and interactive assistance for day tour planning. In *Information and Communication Technologies in Tourism 2011 - Proceedings of the International Conference in Innsbruck, Austria, January 26-28, 2011*, pages 25–37, 2011.

[24] Yohei Kurata and Tatsunori Hara. Ct-planner4: Toward a more user-friendly interactive day-tour planner. In *Information and Communication Technologies in Tourism 2014 - Proceedings of the International Conference in Dublin, Ireland, January 2014*, pages 73–86, 2014.

[25] Jovin J. Mwemezi and Youfang Huang. Optimal facility location on spherical surfaces: Algorithm and application. *New York Science Journal*, 2011.

[26] Alejandro Torreño Oscar Sapena, Eva Onaindia. Flap: Applying least-commitment in forward-chaining planning. *AI Communications*, 28(1):5–20, 2015.

[27] BResnick P. and Varian H. Recommender systems. *Communications of the ACM*, 40, 1997.

[28] BResnick P. and Varian H. Personalised hypermedia presentation techniques for improving online customer relationships. *The Knowledge Engineering Review*, 16, 2001.

[29] Burke R. *The Adaptive Web.* 2007.

[30] Arturo Montejo Ráez, José M. Perea-Ortega, Miguel Angel García Cumbreras, and Fernando Martínez Santiago. Otiŭm: A web based planner for tourism and leisure. *Expert Syst. Appl.*, 38(8):10085–10093, 2011.

[31] Fernando Martínez Santiago, Francisco Javier Ariza-López, Arturo Montejo Ráez, and Luis Alfonso Ureña López. Geoasis: A knowledge-based geo-referenced tourist assistant. *Expert Syst. Appl.*, 39(14):11737–11745, 2012.

[32] Konstan J.A. Sarwar B.M., Karypis G. and Riedl J. Analysis of recommendation algorithms for e-commerce. *ACM Conference on Electronic Commerce*, 2000.