Departamento de Informática

PROYECTO FIN DE CARRERA

Desarrollo y Selección de Características basadas en Distancias entre Grafos para Problemas de Predicción en Planificación Automática

Autor: Isabel Rosario Cenamor Guijarro

Tutor: Fernando Fernández Rebollo
Co-director: Tomás de la Rosa Turbides

Leganés, Septiembre de 2011

Titulo: Desarrollo y selección de características basadas automática
Autor: Isabel Rosario Cenamor Guijarro
Tutor: Fernando Fernández Rebollo
Co-director: Tomás de la Rosa Turbides

EL TRIBUNAL

Presidente: Daniel Borrajo Millán

Vocal: David Expósito Singh

Secretaria: Raquel Fuentetaja Pizán

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 30 de Septiembre de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Cuando un ciclo termina con éxito, siempre cabe recordar todos los momentos que hemos vivido y a todas las personas, gracias a las cuales, se ha conseguido llegar al final. Terminar una carrera es un largo camino no carente de dificultades, que con el apoyo necesario es posible terminar.

En primer lugar quería agradecérselo a mi familia, que siempre me ha ayudado cuando ha podido y me ha ofrecido todo su apoyo.

En segundo lugar, me gustaría agradecer a todos los profesores que me han dado clase, ya que siempre se aprende algo útil con cada uno de ellos y con ellos he llegado a tener el conocimiento que hoy poseo. En especial me gustaría nombrar a mi tutor y al co-director de este proyecto, sin ellos no hubiese sido posible terminar.

En último lugar, y no por ello menos importante, quiero agradecer todo el apoyo que me han brindado mis amigos. Ellos son el bastón que me ha llevado a terminar este camino; sin ellos, esto no sería posible. Dos de ellos han caminado parejo en el último tramo del camino, sufriendo conmigo las dificultades que pueden llegar a existir cuando tienes que estudiar, a ellos muchas gracias. Y sobre todo a Carlos por ser el sol que hace que me levante día tras día. Simplemente: ¡Gracias!

Resumen

El objetivo del presente proyecto es el desarrollo y la selección de características basadas en la distancia entre grafos para problemas de predicción en planificación automática.

En la planificación automática existen dos componentes: los dominios y los problemas. De estos elementos se puede generar diversa información, como el número de objetos, número de predicados, la longitud de los planes que los resuelven o el número de nodos generados en el proceso de cómputo del plan. Además de esta información se va a utilizadar una serie de grafos que permiten comparar distintos problemas. Estos grafos se basan en la representación SAS+, que consiste en una lógica proposicional que posee variables de estado multivaluadas con el fin de expresar una estructura causal subyacente.

Con todos los datos que se han obtenido, se ha creado una representación común independiente del dominio de planificación. Después se van a realizar una serie de comparaciones entre el problema a resolver y la base de hechos (problemas con solución), creando para ello unas medidas de distancia para los grafos. Estas comparaciones han sido almacenadas para su posterior recuperación.

Todo el conocimiento generado se puede utilizar para generar modelos de predicción o para el razonamiento basado en casos. Y con estos modelos se ha comprobado si las medidas desarrolladas con los grafos son acertadas.

Además se han realizado procesos de selección de características de los datos de los problemas, para concluir qué variables son las más relevantes para los problemas de planificación automática.

Las principales conclusiones obtenidas de este proyecto son la aportación de las características más significativas, además de comprobar que la medida de distancia de los grafos es acertada para la creación de los modelos.

Palabras clave: planificación, SAS+, grafos, medidas de distancia.

Abstract

The objective of this project is to explain the development and selection of features based on the distance between graphs for prediction problems in automated planning.

In automatic planning, there are two fundamental components, the domains and problems. Of these elements can generate various information such as the number of objects, number of predicates, the length of the plans that meet or the number of nodes generated in the process of computing the plan. Addition to this information, we used a series of graphs. These graphs are based on the representation SAS+, which is a propositional logic that has multi-valued state variables in order to express an underlying causal structure.

With all the gathered data, a domain-independent representation has been developed. Then they will make a series of comparisons between two different problems, thus creating some distance measures for graphs. These comparisons have been stored for later retrieval.

All the knowledge generated can be used to generate predictive models or case-based reasoning. And with these models, wether the measures developed in the graphs are correct has been checked.

In addition, feature selection processes have been developed, to conclude which variables are relevant to the problems of automatic planning.

The most important conclusions from this project are providing the most significant features and to verify that the graph based distance measure are accurate for creating prediction models.

Keywords: planning, SAS +, graphs, distance measurements.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción y objetivos

En este capítulo se explican los objetivos del proyecto actual así como una breve introducción al mismo y la estructura de la memoria.

1.1. Introducción

A lo largo de los años se han creado muchos planificadores. Todos ellos han utilizado distintas heurísticas para conseguir alcanzar sus objetivos, intentando, en cada ocasión, mejorar los resultados anteriormente conseguidos.

Cada planificador utiliza un lenguaje de formalización, siendo los más comunes PDDL y STRIPS, basados en lógica de predicados. Sin embargo, existen otros tipos de formalizaciones como es el caso de SAS+. Ésta es la formalización que se emplea en este proyecto y su elección es debido a que posee un conjunto de grafos que representan el problema, y que permiten hacer comparativas entre problemas basadas en la comparativa de los grafos que los representan.

En la planificación existen dos componentes fundamentales: el dominio y el problema. En la formulación SAS+, una parte fundamental de su formulación se basa en la representación del problema mediante grafos, como se ha comentado anteriormente. Incluyendo estos datos y otros relevantes de cada problema como son el número de objetos, número de literales en el estado inicial y final, el primer valor de la heurística, etc. se ha creado una estructura de problema.

Además se han necesitado datos relacionados con la solución de un problema (plan). Para conseguir estos datos se ha necesitado la ayuda de un planificador. Después de la realización de un estudio de los planificadores del mercado y ver cuales son las características, se ha seleccionado el planificador LAMA por ser uno de los más recientes planificadores que usan la formulación SAS+. Además este planificador posee un traductor que convierte un problema formulado en PDDL en SAS+ y lo transforma en una representación interna.

Los ficheros LAMA han sido utilizados para la creación de los ficheros post-procesado que contienen toda la información que se ha extraído de los anteriores, es decir, todos los datos de entrada antes comentados y los datos de salida, como son la solución del problema (el plan), la longitud del mismo, el número de nodos evaluados y generados, el tiempo total que tarda en encontrar la solución y el tiempo total del planificador; es decir, todos menos los relacionados con los grafos.

Cuando se han obtenido al menos dos ficheros (dos problemas) se pasan a la fase de comparación. Esta fase consiste en la comparación de todos los atributos. Para los atributos relacionados con los grafos, se han creado unos algoritmos para conseguir una medida de distancia entre grafos que tenga un valor significativo para cada problema. Estos datos han sido almacenados en un conjunto de ficheros que se denominan de post-comparación.

Con el conjunto de datos post-comparación no se puede obtener una información relevante a simple vista. Para ello, se tienen que transformar los datos a un formato que por si sólo nos ayude a obtener la información o por el contrario, a un formato preestablecido de una aplicación existente que nos ayude a obtener la información. Se ha seguido la segunda opción y el programa elegido para el análisis de datos es Weka.

Para realizar un análisis, en primer lugar se necesita saber qué es lo que queremos conseguir. Para ello se ha realizado un estudio en detalle de los datos aportados hasta el momento, dando como resultado que las variables de salida que determinan la longitud y el número de nodos evaluados de un problema son los más relevantes. En función de estos resultados se han creado varios conjuntos de entrada para determinar cual es el mejor.

Los conjuntos de entrada han sido divididos en función del dominio y una función objetivo. En este proyecto nos planteamos predecir los valores de salida: longitud de la solución y número de nodos evaluados. Además dentro de esta división, se ha incluido un filtro de selección de atributos y otro que elimina los datos fuera de rango (*outlayers*).Todo este conjunto de datos compone un experimento de Weka.

En cada experimento se han pasado una serie de algoritmos para la obtención de modelos y saber qué fichero de datos es el que tiene mejor resultado. Una vez obtenido el mejor fichero, se ha analizado qué atributos se encuentran en ese fichero para determinar cuáles son los más relevantes en los problemas de planificación automática.

Con este análisis, además, se quiere conseguir unos modelos que aproximen los valores de salida con el menor error posible, pudiendo ser aplicados posteriormente en la construcción de un planificador basado en instancias, o para evaluar la complejidad de un problema.

1.2. Objetivos

El objetivo de este proyecto consiste en la construcción de un sistema capaz de extraer la similitud entre dos problemas comparando todos los elementos posibles del mismo para, finalmente, extraer las características más relevantes de los problemas de planificación automática con técnicas de análisis de datos. Para ello se ha sacado la información del planificador LAMA y se ha procesado para ofrecer un formato independiente de ese planificador. La selección del planificador LAMA se ha debido a que se trata de uno de los últimos planificadores creados que utiliza la formulación SAS+. Parte de los datos sacados de LAMA lo conforman un conjunto de grafos que representan un problema. Para poder comparar los problemas a partir de esos grafos se han implementado distintas medidas de similitud. Cada una de estas medidas aporta la comparación de los grafos de dos problemas, como es la comparación del *domain transition graph* y del *causal graph* que se descubrirán posteriormente. Además de estas dos me-

didas, se comparan los estados iniciales del segundo grafo. Una vez conseguido un conjunto de datos se han transformado a formato Weka y se han dividido en distintos grupos para saber si la medida de similitud aportada es relevante para predecir atributos de salida en los problemas de planificación.

En base a este objetivo principal, se proponen los siguientes objetivos específicos:

- Estudio de los distintos conceptos claves desarrollados en el capítulo 2, para ampliar el conocimiento sobre el tema. De esta manera, se hace más sencillo abordar el desarrollo del proyecto. Además, se ha realizado un estudio del estado del arte actual, revisando de esta manera herramientas de planificación, estudiando en profundidad el planificador LAMA, dado que este planificador utiliza la formulación SAS^+ y es usado en la parte de creación de los ficheros de datos. Además se ha recopilado información de grafos, así como de técnicas de análisis de datos.
- La obtención de ficheros de datos de post-procesado independientes del planificador LAMA que contengan toda la información relevante de los problemas.
- Creación de distintas medidas para la obtención de la medida de distancia de los grafos para la comparación de los problemas.
- Creación de ficheros de salida con los resultados obtenidos de la comparación de problemas.
- Creación de distintos experimentos en función de los dominios y la función de evaluación.
- Selección de distintas técnicas de análisis de datos para la obtención del modelo.
- Análisis cuantitativo y cualitativo de los resultados ofrecidos por los modelos.

1.3. Estructura de la memoria

La presente memoria se ha dividido en 8 capítulos, cuyo contenido se detalla a continuación.

El presente capítulo se corresponde con la introducción del proyecto. Se explica la motivación que ha llevado a la implementación del sistema, los objetivos que se persiguen con su diseño, los medios empleados y la estructura del documento.

En el Capítulo 2, se explican los conceptos teóricos relacionados con la planificación, centrándose en la planificación SAS^+. En primer lugar se describe el concepto de planificación y distintos planificadores del mercado, centrándose, en el planificador LAMA. Después se explican los dominios empleados en este proyecto, para continuar con la formulación STRIPS y PDDL. En el siguiente punto del capítulo se explican los métodos de búsqueda, para a continuación explicar en detalle la formulación SAS+ con cada una de sus partes: representación SAS+, definición de una tarea y subtarea SAS+ y planificación heurística basada en *causal graph*. Posteriormente se detalla el concepto de grafo y sus medidas

de distancias y para finalizar una breve explicación de las técnicas de análisis de datos.

En el Capítulo 3, desarrollo del proyecto, se exponen los distintos procesos que se han seguido para construir el sistema. Este proceso se inicia con la creación de unos ficheros donde se guardan las estructuras necesarias de cada uno de los elementos que conforman un problema. Estas estructuras plasman todas las características de los problemas y las medidas creadas de comparación de grafos.

En el Capítulo 4, experimentación, se lleva a cabo un análisis de datos con tres dominios elegidos para conseguir las relaciones entre los distintos atributos de los problemas. Las técnicas empleadas en este apartado se explican en el apartado 2 de conceptos básicos.

El Capítulo 5 está dedicado a la gestión del proyecto. En él se especifican las fases del desarrollo del proyecto, su planificación, los medios empleados y el presupuesto para su realización.

En el Capítulo 6, conclusiones, se presentan las conclusiones obtenidas durante el desarrollo y evaluación del sistema.

En el Capítulo 7, trabajos futuros, se presentarán los posibles desarrollos futuros para mejorar el funcionamiento o posibilidades de ampliar el desarrollo del mismo.

Al final del documento se podrán encontrar tres anexos, con formatos de salida y entra empleados, el glosario de términos, un breve manual de usuario, el cual es útil para entender el funcionamiento del sistema, así como los parámetros necesarios para su correcto funcionamiento. Por último se incluye la bibliografía.

Capítulo 2

Estado de la cuestión

En este capítulo se van a introducir los conceptos básicos para entender el desarrollo del proyecto. Estos van desde conceptos de planificación, de teoría de grafos y de análisis de datos.

2.1. Planificación

La Planificación Automática es una rama de la Inteligencia Artificial que se refiere a la generación automática de planes compuestos por secuencias de acciones, normalmente para ser ejecutadas por agentes inteligentes, robots o vehículos autónomos no tripulados. Esta área de la Inteligencia Artificial surgió hacia 1971, fecha en la que nace el planificador STRIPS [FN72]. STRIPS fue creado con la finalidad de controlar el robot Shakey y conseguía que éste resolviese distintos problemas. Los programas de planificación que incorporan estos algoritmos se denominan planificadores [NGT04].

Un planificador típicamente toma tres entradas: una descripción del estado inicial del mundo, un conjunto de acciones posibles y una descripción de las metas deseadas. Las acciones constituyen el dominio de planificación, mientras que la descripción compone el estado inicial y las metas forman la parte del problema de planificación a resolver.

Todo planificador produce una secuencia de acciones que van desde el estado inicial a un estado de cumplimiento de la meta. Generalmente, cada acción especifica precondiciones que se deben cumplir como requisito para ejecutar la acción, así como postcondiciones, que constituyen el efecto de la acción sobre el estado actual.

Existen varias formas de planificar y estas pueden ser: en función de los fines (metas) y los medios (operadores), descomponiendo el problema en subproblemas (jerárquicamente), basado en la experiencia, reactivamente, entre varios agentes, estableciendo prioridades, etc.

Para resolver problemas de planificación es necesario formalizarlos. La forma más empleada es la lógica de predicados. La lógica de predicados permite representar las cuestiones ciertas o falsas del mundo (problema a planificar) mediante: términos, predicados, conectivas, y cuantificadores.

El lenguaje más común es *Planning Domain Definition Language* (PDDL) [FL03], que se encuentra basado en STRIPS, donde las tareas se encuentran divididas

en dominio y problemas.

La dificultad de la planificación depende de los supuestos simplificadores empleados, tales como el tiempo discreto, el determinismo del entorno y omnisciencia. Los planificadores clásicos, que realizan todas esas simplificaciones, se han estudiado plenamente. Algunas de las técnicas más populares incluyen el encadenamiento hacia adelante (*forward chaining*) y encadenamiento hacia atrás (*backward chaining*) explicados en el punto 2.5

A continuación se detallan los tipos más comunes de planificación.

- **Planificación de preferencia basada en el objetivo**. No es sólo para producir un plan, sino también para satisfacer las preferencias especificadas por el usuario [BM09, SP06].

- **Planificación condicional.** Opera en un SI (condición) ENTONCES (acción) SI NO (acción). El modelo de bifurcación no tiene que ser binario. Planificadores condicionales examinan el modelo actual y determinan la acción a tomar en base a eso. Este proceso permite que el algoritmo de planificación examine todas las posibles contingencias y luego realiza la mejor acción posible, dada la situación actual. La planificación condicional es, sin embargo, limitada debido a que si el número de condiciones es lo suficientemente grande el problema rápidamente puede convertirse en intratable. Además, los planificadores condicionales sólo pueden actuar sobre las condiciones específicamente enumeradas [BV97, PS92].

- **Planificación continua**. Opera mediante la realización de un cierto número de acciones, analiza del estado actual del mundo, y luego reevalúa sus objetivos. La planificación continua permite revisar los objetivos, con el fin de satisfacer las necesidades actuales en el proceso de alcanzar su objetivo final. Por ejemplo, un algoritmo de planificación continua puede decidir que con el fin de alcanzar su objetivo, primero se debe planificar la realización de un subobjetivo. Este proceso de planificación permite a los planificadores continuos funcionar de forma autónoma, ya que se basan en la suposición de un mundo cerrado [GFG$^+$97, ERMC00].

2.2. Planificadores

En este punto se van a describir las principales características de varios planificadores, para acercar al lector un poco más a la idea de lo que es un planificador.

2.2.1. GRAPHPLAN

Graphplan es un planificador basado en grafos que surgió en 1995 [Blu95]. Graphplan construye una representación compacta del espacio de estados con un grafo de planificación y después hace búsqueda sobre él. Este grafo constituye una representación compacta del espacio de estados porque considera como un único nodo todos los estados que se generan en el mismo nivel haciendo una búsqueda hacia delante.

El grafo de planificación tiene dos tipos de nodos que forman niveles consecutivos: los nodos de proposición y los nodos de acción. Los nodos de proposición

contienen todos los efectos de las acciones del nivel de acciones previo. Los nodos de acción contienen todas las acciones cuyas precondiciones están presentes en el nivel de proposiciones (nivel previo). El primer a nivel de proposiciones contiene todas aquellas proposiciones que son ciertas en el estado inicial. Graphplan trabaja con todas las acciones instanciadas, por lo que requiere que las acciones se instancien en una etapa de preproceso [LB03].

2.2.2. FF

FF [HN01] es un planificador totalmente independiente del dominio que se basa en buscar hacia delante en la espacio de estados, guiado por una heurística que estima las distancias meta haciendo caso omiso "las listas borradas". Fue desarrollado por Jörg Homann. Es un sistema implementado íntegramente en *C* y participó en la tercera IPC mostrando un comportamiento muy competitivo.

Metric-FF es el sucesor de FF. Tiene sus mismas características pero permite utilizar variables de estado numéricas.

2.2.3. AltAlt

AltAlt [NNK00] de Rao Kambhampati, es un planificador independiente del dominio sobre la base de una combinación de Graphplan y tecnología heurística de búsqueda de espacio de estados. La primera versión de AltAlt compitió en el concurso internacional de planificación (AIPS-2000 [Bac01]). Aunque esta versión no fue depurada por completo, obtuvo resultados similares al resto de los planificadores. Este es un planificador híbrido que utiliza representación gráfica de Graphplan para obtener una familia de heurísticas de búsqueda muy eficaz, y estas son usadas para impulsar la búsqueda de estados en el planificador.

La nueva versión de AltAlt ha sido mejorada con una serie de optimizaciones para reducir el coste computacional en la búsqueda de heurísticas, así como la búsqueda de estados. En concreto, se controla el uso parcial de gráficos de planificación para el coste de la computación en la heurística y subconjuntos de acciones en el gráfico de la planificación para limitar el factor de ramificación de la búsqueda de regresión.

2.2.4. SAPA

El planificador SAPA [DK03] fue construidó con un modelo de planificación que combina la planificación basada en costes y la planificación temporal. Utiliza el algoritmo de búsqueda mejor primero cuya función de evaluación es sensible tanto a los costes como a la duración total del plan considerando acciones concurrentes (*Makespan*). El espacio de estados en el caso de SAPA se compone de estados que incluyen información relevante para la planificación temporal.

2.2.5. MO-GRT

El planificador MO-GRT [GSS04] es una versión del planificador GRT [RV11] que es capaz de considerar a la vez varios criterios para evaluar la calidad de los planes. GRT representa problemas STRIPS independientes de dominio en dos fases. En la fase de preprocesamiento, se estima la distancia entre cada hecho y de los objetivos del problema, en búsqueda hacia atrás. Luego, en la fase de

búsqueda, estas estimaciones se utilizan para estimar aún más la distancia entre cada estado intermedio y las metas, esto se realiza con búsqueda hacia delante y el mejor primero. El cálculo de los estimadores se basa en conjuntos de vectores que se asignan a los hechos del problema, y permiten estimar el coste total de conseguir ciertos hechos a partir de las metas (la heurística se calcula de forma regresiva).

2.2.6. LAMA

LAMA [RW08, RW10, Ric10] es un sistema de planificación proposicional basado en heurísticas de búsqueda. Usa una codificación con variables no binarias, que poseen un conjunto de valores finitos y varias heurísticas para la búsqueda del plan. Este planificador se explica más en detalle porque es necesario para la comprensión de posteriores partes del documento.

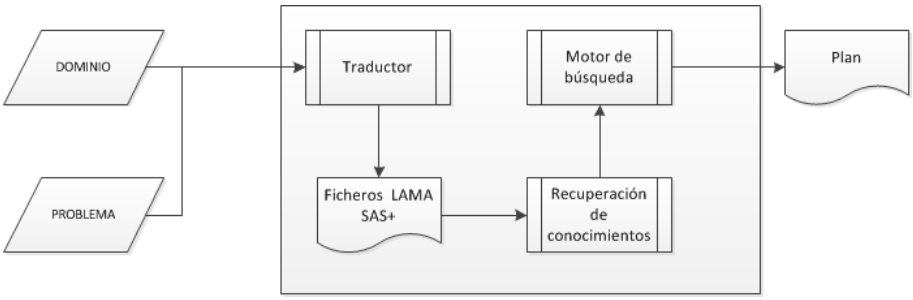


Figura 2.1: Composición LAMA

Este planificador consta de tres partes fundamentales:

- ***El traductor:*** Se encarga de transformar el problema PDDL a SAS^+.
- ***El módulo de recopilación de conocimientos:*** Usando la representación generada por el traductor, este módulo es responsable de construir una serie de estructuras de datos que tienen un papel principal en la generación de etiquetas y posteriormente en la búsqueda. Por ejemplo, el *domain transition graph* (posteriormente explicado) codifica cada variable de estado con todos sus posibles valores que pueden intercambiarse a través de sus operadores. Por otra parte este módulo construye estructuras de datos para determinar el conjunto de acciones aplicables a un determinado estado de planificación y evaluar los valores derivados de las variables.
- ***El motor de búsqueda*:** Usando las estructuras anteriormente mencionadas, busca un plan utilizando heurísticas mejoradas. Usa preferencia en los operadores (similar a las acciones de ayuda de FF) y evaluación heurística diferida que mitiga el impacto de tareas con alto factor de ramificación.

Para el entendimiento de este proyecto es necesario entrar en más profundidad en las estructuras que se han creado en el traductor. Estas estructuras se guardan en ficheros externos (ficheros LAMA SAS^+ en la figura 2.1). Y el contenido de las mismas es el siguiente:

1. **Variables del sistema (variables de estado)**: Todas las variables de los problemas pueden tener de 2 a n valores. Normalmente estas variables están asociadas a algún concepto del sistema. Si elegimos el problema *logictics*, una variable está destinada a un paquete, otra a un camión, otra a un avión, etc.

2. **Dominio de las variables**: Nos dice cuantos valores tienen cada una de las variables. Se incluye el valor -1, para indicar que la variable no tiene asignado ningún valor.

3. **Estado inicial**: Nos dice el valor inicial en el que se encuentra cada variable. No es un valor textual, que es el que verdaderamente tienen, sino el número de orden que tienen en la lista de valores de cada variable.

4. **Estado final**: Nos dice que variables y sus valores son necesarios para alcanzar el objetivo del problema.

5. ***Domain Transition Graph*** **(DTG)**: Grafos de cada una de las variables de estado. Para ello utilizan los operadores instanciados anteriormente descritos, para describir las transiciones de los mismos. Cabe mencionar que todos los operadores tienen que ser utilizados en alguno de los grafos de esta sección.

6. ***Causal graph*** **(CG)**: Gráfico único que relaciona las variables de estado. Esta relación reside en el número operadores que hay que utilizar para llegar desde el estado inicial al final. Por ello siempre están unidas las variables que no representan parte del objetivo con las que sí lo son.

7. **Lista de operadores instanciados**: Para la creación de cada uno de los DTG del problema, LAMA necesita crear todos los operadores. Para ello lo que hace es, a partir de los operadores originales del problema, crear un operador instanciado con cada una de las posibilidades que puedan ocurrir. Estos operadores pueden tener *pre* y *post* condiciones.

Además existe una estructura que almacena valores que nunca se pueden dar al mismo tiempo para evitar callejones sin salida en el planificador.

2.3. Dominios empleados

Los dominios empleados de este proyecto son tres: *logitics*, *parking* y *storage*. Y para conseguir un buen entendimiento de los resultados, es necesario explicar en qué consiste cada uno de ellos.

2.3.1. *Logistics*

Este dominio consiste en el transporte de 1 a n paquetes a un destino que se encuentre dentro de un mapa.[RBV01]

Los elementos de los que consta este dominio son: lugar, ciudad, avión, paquete, aeropuerto y camión.

Los predicados que existen en este dominio son:

- $(in - city ?loc - place ?city - city)$: Nos dice que un lugar $?loc$ es de una ciudad $?city$.

- $(at?obj - physobj?loc - place)$: Un objeto $?obj$ (avión, camión, aeropuerto) está en un lugar $?loc$.
- $(in?pkg - package?veh - vehicle)$: Un paquete $?pkg$ se encuentra dentro de un vehículo $?veh$ (avión, camión).

Las acciones de este dominio son las que aparecen en la tabla 2.1:

Nombre de la acción	movimiento
Load - truck	Carga un paquete en un avión.
Load - airplane	Carga un paquete de un camión.
Unload - truck	Descargar un paquete de un camión.
Unload - airplane	Descarga un paquete de un avión.
Drive - truck	Desplazamiento del camión entre dos lugares.
Fly - airplane	Desplazamiento del avión entre dos lugares.

Cuadro 2.1: Acciones de *logistics*

En la imagen 2.2 se muestra un ejemplo de problema *logistics*. Consta de dos camiones, dos aviones, tres aeropuertos, seis lugares, dos ciudades y un paquete. El estado inicial del paquete es B, y el estado final es F.

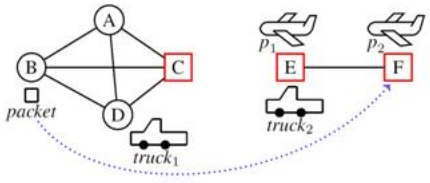


Figura 2.2: Ejemplo de problema *logistics*

2.3.2. *Parking*

Este dominio consiste en alcanzar una colocación determinada de unos coches dentro de un *parking* con la característica que en cada posición (o bordillo) pueden estar dos coches aparcados en doble fila.

Los elementos de este dominio son: coche y bordillo.

Los predicados de este dominio son:

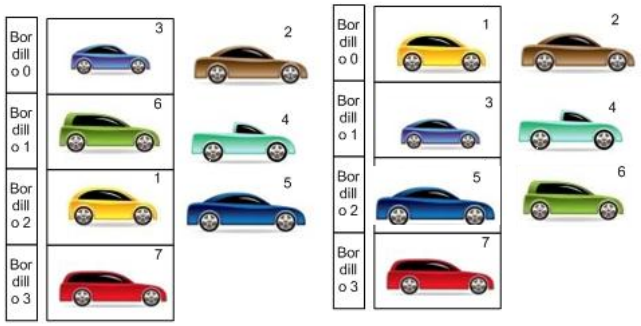
- $(at - curb?car - car)$: El coche $?car$ se encuentra al lado de un bordillo.
- $(at - curb - num?car - car?curb - curb)$: El coche $?car$ se encuentra en el bordillo $?curb$.
- $(behind - car?car?front - car - car)$: El coche $?car$ se encuentra detrás de otro coche $?front - car$ (en doble fila).
- $(car - clear?car - car)$: Detrás del coche $?car$ no hay otro coche.
- $(curb - clear?curb - curb)$: Este bordillo $?curb$ está vacío.

Las acciones de este dominio son las siguientes que aparecen en la tabla 2.2:

Nombre acción	movimiento
move-curb-to-curb	Consiste en mover de un bordillo a otro bordillo.
move-curb-to-car	Consiste en mover de un bordillo a doble fila.
move-car-to-curb	Consiste en mover de en doble fila a un bordillo.
move-car-to-car	Consiste en mover de en doble dila a doble fila.

Cuadro 2.2: Acciones de *parking*

Un ejemplo de este tipo de problemas es el que se muestra en la imagen 2.3. Este consta de 7 coches y 4 bordillos. El estado inicial es como se encuentra en la sub-figura 2.3a y el estado final en la sub-figura 2.3b.



(a) Estado inicial (b) Estado final

Figura 2.3: Ejemplo de problema *Parking*

2.3.3. *Storage*

Storage [GS02] es un dominio de planificación con el razonamiento espacial. Básicamente, el dominio consiste en pasar un cierto número de cajas de algunos contenedores a unos almacenes en grúas. Dentro de un depósito, cada grúa puede moverse de acuerdo a un mapa espacial de conexión especifica las diferentes áreas de la estación.

Los elementos de este dominio son: contenedores, cajas, grúas áreas y almacenes.

Los predicados de este dominio son:

- $(clear?s - storearea)$: El área $?s$ esta vacía
- $(in?x - storeobject?p - place)$: Un contenedor $?x$ esta en un lugar $?p$.
- $(available?h - hoist)$: Una grúa $?h$ esta disponible.
- $(lifting?h - hoist?c - crate)$: Un paquete $?c$ esta levantado por una grúa $?h$.
- $(at?h - hoist?a - area)$: Una grúa $?h$ esta en una área $?a$.

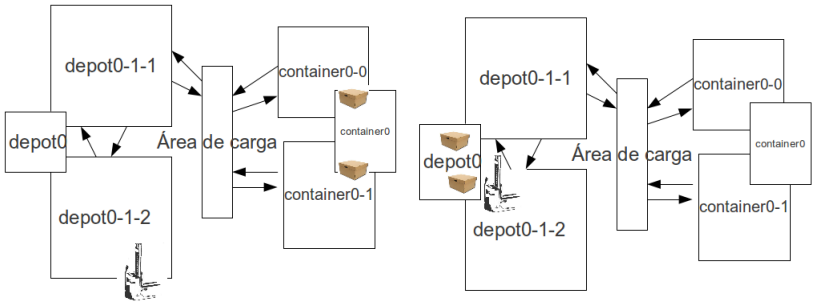
- $(on ?c - crate ?s - storearea)$: Una caja $?c$ esta en un almacén $?s$.
- $(connected ?a1 ?a2 - area)$: Dos áreas $?a1$, $?a2$ están conectadas.
- $(compatible ?c1 ?c2 - crate)$: Dos cajas $?c1$, $?c2$ son compatibles.

Las acciones de este dominio son las siguientes que aparecen en la tabla 2.3:

Nombre acción	movimiento
lift	Una grúa eleva una caja de posición en un área del almacén.
drop	Una grúa deja una caja de posición en un área del almacén.
move	Mover una grúa de un almacén a otro.
go-out	Sacar una grúa del almacén.
go-in	Meter una grúa al almacén.

Cuadro 2.3: Acciones de *storage*

Un ejemplo de este tipo de problemas es el que se muestra en la imagen 2.4. Este consta de 4 áreas y 4 bordillos. El estado inicial es como se encuentra en la sub-figura (a) y el estado final la sub-figura (b).



(a) Estado inicial (b) Estado final

Figura 2.4: Ejemplo problema *storage*

2.4. STRIPS

En la inteligencia artificial, STRIPS [FN72] (*Stanford Research Problem Solver Institute*) es un planificador automático desarrollado por Richard Fikes y Nils Nilsson en 1971. El mismo nombre fue utilizado más adelante para referirse a la lenguaje oficial de las entradas a este planificador. Este lenguaje de cuarta generación[1] es la base para la mayoría de los planificadores para expresar el problema de la planificación automática.

[1]Son lenguajes que se relacionan menos con procedimientos y que son aun más parecidos al lenguaje natural que los anteriores. Entre sus características se incluyen las capacidades de consulta y base de datos, de creación de códigos y capacidades gráficas.

2.4.1. Representación STRIPS

El uso de la representación STRIPS [Byl94] permite utilizar heurísticas independientes del dominio. Por ejemplo, dado un estado, la cantidad de objetivos que aún no se han satisfecho. Además, la búsqueda de planes hacia delante permite generar los sucesores a partir del estado inicial. Esta búsqueda es completa (el número de objetos es finito) pero ineficiente en la práctica. Sin embargo, algunos problemas persisten, como el gran factor de ramificación o el que la búsqueda del plan se realice de manera secuencial.

Otro tipo de búsqueda es la búsqueda hacia atrás. En este caso, se comienza por el objetivo, en cada estado se generan los posibles predecesores y se finaliza cuando se alcanza un objetivo cierto en el estado inicial. Este sistema es más eficiente que el anterior, debido a que tiene un menor factor de ramificación.

2.4.2. Definición

Una instancia de STRIPS se compone de:

- Un estado inicial.
- La especificación de los estados meta - situaciones que el planificador está tratando de alcanzar.
- Un conjunto de acciones. Para cada acción, se incluyen los siguientes:
  - Condiciones previas (lo que debe ser establecido antes de la acción se lleva a cabo).
  - Post-condiciones (lo que se estableció después de la acción se lleva a cabo).

Matemáticamente, una instancia de STRIPS es un cuádruple (P, O, I, G), en el que cada componente tiene el siguiente significado:

- P es un conjunto de condiciones (es decir, variables, proposicional)
- O es un conjunto de operadores (por ejemplo, acciones), cada operador es en sí misma otro conjunto de cuatro elementos $(\alpha, \beta, \gamma, \delta)$, siendo cada elemento de un conjunto de condiciones. Estos cuatro conjuntos son:
  - α Conjunto de condiciones que se deben cumplir para que se ejecute la acción.
  - β Conjunto de condiciones que deben ser falsas para que se ejecute la acción.
  - γ Conjunto de hechos que son verdaderos cuando ocurre la acción.
  - δ Conjunto de hecho que son falsos cuando ocurre la acción.
- I es el estado inicial, es decir, el conjunto de condiciones que inicialmente son verdaderas (todos los demás se supone que son falsas).
- G es la especificación del estado de la meta, lo que se da como un par (N, M), que especifica que las condiciones son verdaderas y falsas, respectivamente, a fin de que un estado sea considerado como un estado objetivo.

Un plan para una instancia de planificación es una secuencia de operadores que se pueden ejecutar desde el estado inicial y que conduce a un estado objetivo.

Formalmente, un estado es un conjunto de condiciones: un estado está representado por el conjunto de condiciones que son verdad en él. Las transiciones entre los estados se modelan mediante una función de transición, que es una función de mapeo que cambia los estados en función de la ejecución de acciones. Dado que los estados están representados por un conjunto de condiciones, la función de transición con respecto a la instancia de STRIPS (P, O, I, G) se trata de una función

$$succ : 2^P \times O \to 2^P$$

donde 2^P es el conjunto de todos los subconjuntos de P, y por lo tanto el conjunto de todos los estados posibles.

La función de transición *succ* de un estado $C \subseteq P$, se puede definir de la siguiente manera, con el supuesto simplificador de que las acciones siempre se puede ejecutar, pero no tienen efecto si sus condiciones no se cumplen:

$succ\,(C, (\alpha, \beta, \gamma, \delta)) = C/\delta \bigcup \gamma$ si $\alpha \subseteq C$ y $\bigcup \beta \bigcap C = \emptyset$.

La función *succ* se puede extender a las secuencias de las acciones de las ecuaciones de recurrencia siguientes:

$succ(C, []) = C$

$succ(C, [a_1, a_2, \ldots, a_n]) = succ(succ(C, a_1), [a_1, a_2, \ldots, a_n])$

Un plan en STRIPS es una secuencia de acciones desde el estado inicial hasta la meta que resultan de la ejecución. Formalmente,$[a_1, a_2, \ldots, a_n]$ es un plan para $G = (N, M)$ si $F = succ(I, [a_1, a_2, \ldots, a_n])$ satisface las siguientes condiciones:

- $N \subseteq F$
- $M \bigcup F = \emptyset$

2.5. PDDL

El dominio del lenguaje de definición de Planificación (PDDL) es un intento de estandarizar la planificación de dominio y lenguajes de descripción del problema. Fue desarrollado principalmente para la *International Planning Competitions* [TTLY98], por Drew McDermott [GHK+98] en 1998. La última versión de este lenguaje es PDDL3.1.

La planificación de tareas especificadas en PDDL se separan en dos archivos:

Un archivo de dominio para los predicados y las acciones. Un archivo de problema para los objetos, los estados inicial y las especificaciones objetivo.

PDDL también es compatible con las preferencias basadas en la planificación (*preference-based planning*)[2].

[2]La planificación basada en las preferencias es una forma de planificación y programación automatizada que se centra en la elaboración de planes que, además, satisfacer el mayor número de preferencias especificadas por el usuario como sea posible. En los dominios de muchos problemas, una tarea puede ser realizada por varias secuencias de acciones. Estos planes pueden variar en calidad: puede haber muchas maneras de resolver un problema, pero que generalmente se prefiere una forma que sea, por ejemplo, costo-efectiva, rápida y segura.

2.5.1. Dominios en PDDL

Estos ficheros son los que anteriormente se ha comentado que contienen los predicados y las acciones. En la primera línea del dominio se define a que tipo de dominio corresponde. En la siguiente línea los requisitos del problema, en las siguientes líneas los tipos de datos que aparecen en ese problema. Después se detallan los predicados y en último lugar las acciones. En el siguiente ejemplo del dominio *logistics* se pueden ver las distintas partes.

Como se puede observar en el ejemplo, posee un nombre *logistics*. Los tipos de datos que aparecen son vehículos como el camión y el avión, objetos físicos como los vehículos y los paquetes, los lugares como una localización y el aeropuerto y los objetos que son un lugar, una ciudad y un objeto físico. Después los predicados y por último las acciones, antes explicadas en el apartado 2.3.1.

(define (domain logistics)
(:requirements :strips :typing)
(:types truck airplane - vehicle
package vehicle - physobj
airport location - place
city place physobj - object)
(predicates (in-city loc - place city - city)
(at obj - physobj loc - place)
(in pkg - package veh - vehicle))
...)
(:action load-truck ...)
(:action load-airplane ...)
(:action unload-truck ...)
(:action unload-airplane ...)
(:action drive-truck ...)
(:action fly-airplane
parameters (p - airplane s - airport d - airport)
precondition (and (at p s) (not (= s d)))
effect (and (at p d) (not (at p s))))

2.5.2. Problemas PDDL

Estos ficheros son los que contienen los objetos, los estados iniciales y las especificaciones de los objetivos. Lo primero que contiene este fichero es el nombre del problema y el dominio al que pertenece. En las siguientes líneas aparecen los distintos objetos teniendo en cuenta que tienen que tener su tipo. Después de detallar todos los objetos, se encuentran los predicados que son ciertos en el estado inicial. Y en último lugar los predicados que conforman el estado meta. En el siguiente ejemplo del dominio *logistics* se puede comprobar cada una de las partes.

Como se puede observar en el ejemplo existe un objeto de tipo avión *airplane*0, dos objetos de tipo ciudad *city*0 y *city*1, dos objetos de tipo camión *truck*0 y *truck*1, dos aeropuertos *loc0-0* y loc1-0, dos localizaciones como *loc0-1* y *loc1-1* y un solo paquete *p*0.

(define (problem logistics-c2-s2-p1-a1)

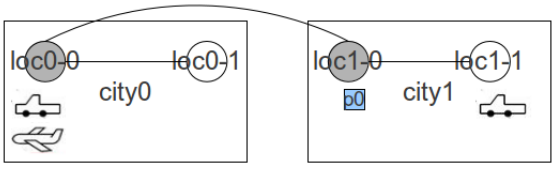
(:domain logistics-strips)
(:objects airplane0 - airplane
city0 city1 - city
truck0 truck1 - truck
loc0-0 loc1-0 - airport
loc0-1 loc1-1 - location
p0 - package)
(:init (in-city loc0-0 city0)
(in-city loc0-1 city0)
(in-city loc1-0 city1)
(in-city loc1-1 city1)
(at truck0 loc0-0)
(at truck1 loc1-1)
(at p0 loc1-0)
(at airplane0 loc0-0))
(:goal (and (at p0 loc0-0))))

2.6. Métodos de búsqueda

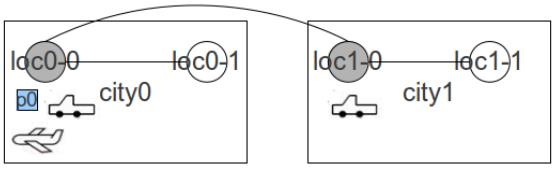
En este apartado se detallan en líneas generales los métodos de búsqueda.

2.6.1. Búsqueda hacia adelante

La búsqueda hacía delante se trata de una progresión, es decir, se ejecutan operadores hasta que se encuentre solución. Ejemplos de planificadores que usan la búsqueda hacía delante: SOAR [LRN86], FF [HN01], TLPLAN [BK00]. Para entender este procedimiento se va a mostrar un ejemplo del dominio *logistics* en la figura 2.5.



(a) Situación inicial



(b) situación final

Figura 2.5: Situación inicial (a) y final (b)

Tenemos de estado inicial, el paquete p_0 se encuentra en la localidad loc1-0. Existen varias acciones posibles: $drive-truck(truck1, loc1-1, loc1-0)$, permite mover el camión de la posición *loc1-1* a loc1-0. Otra acción es $fly(airplane0, loc0-$

$0, loc1 - 0)$, que consiste en el vuelo del avión de una ciudad a otra. En la figura 2.6 se muestra el árbol resultante de la búsqueda hacía delante.

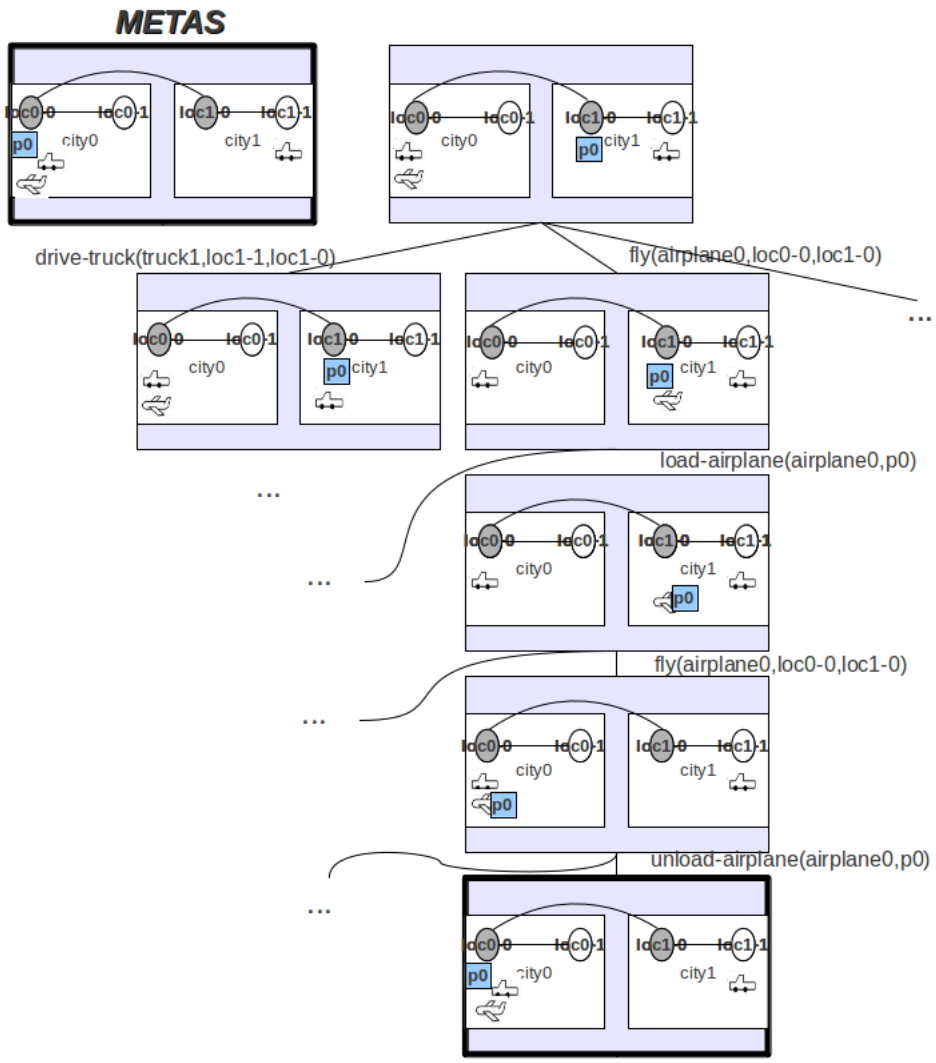


Figura 2.6: Árbol de búsqueda hacía delante - dominio *logistics*

2.6.2. Búsqueda hacia atrás

La búsqueda hacia atrás se trata de una regresión, comienza desde las metas, seleccionando los operadores que las añadan y se añaden las precondiciones de los mismos al conjunto de metas. Ejemplos de planificadores que usan la búsqueda hacía detrás: STRIPS [FN72], UCPOP [PW92], PRODIGY [VCP+95]. Para entender este procedimiento se va a mostrar un ejemplo. Tenemos el mismo ejemplo que en el apartado anterior. A continuación en la figura 2.7 en la página siguiente se muestra el árbol de búsqueda resultante.

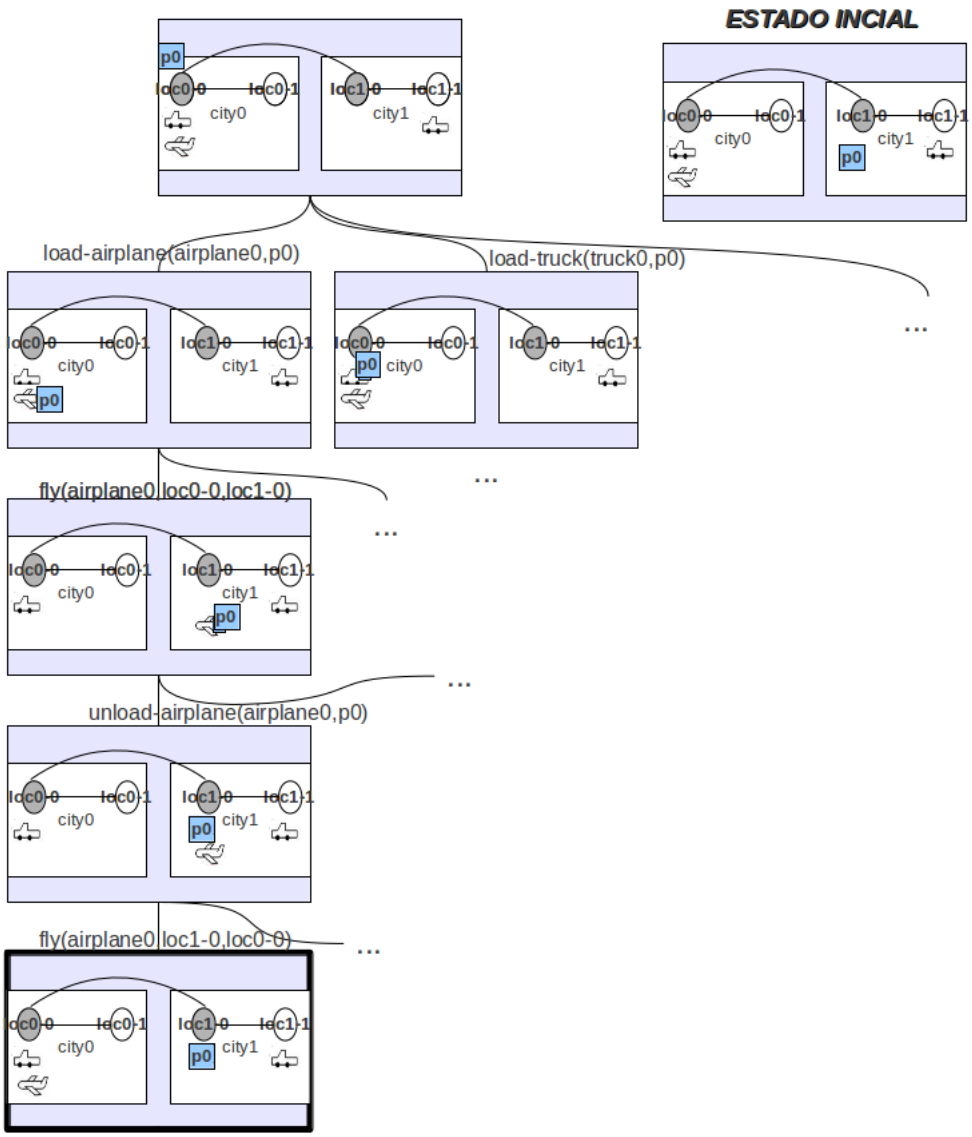


Figura 2.7: Árbol de búsqueda hacía atrás - dominio *logistics*

2.7. Búsquedas informadas

Estas poseen información adicional, en forma de heurística. Estas heurísticas usan un conocimiento especifico del problema y pueden resolver problemas más eficientemente. Siendo el origen de estas, el mismo problema relajando condiciones [PW92].

La relajación se obtiene eliminando condiciones, es decir, precondiciones o postcondiciones [BG98]: dado un problema como $P = (O, I, G)$, la relajación P' de P se define como $P' = (O', I, G)$ donde:

$$O' = \{(pre(o), add(o), \emptyset) \,|(pre(o), add(o), del(o))\epsilon O\}$$

Una secuencia del plan de acciones es un plan relajado si es una solución del problema relajado P' del problema original P.

- Cuanto más parecido sea P' a P, más informada será la función heurística, $h(\cdot)$.

- Cuanto más simplificado sea P' , más fácil será computar $h(\cdot)$.

2.8. Representación SAS^+.

La planificación formal de SAS^+ [Bac92, BN93] es una alternativa sobre la lógica proposicional de STRIPS. A diferencia de STRIPS, SAS^+ permite que las variables de estado tengan dominios finitos, a diferencia de los binarios en STRIPS.

En STRIPS la codificación de la figura 2.8 usaría un total de 18 proposiciones para describir la localización de los camiones, y dos variables más para saber si hay un paquete dentro del camión. Sin embargo, la codificación SAS^+ sólo usa tres variables, dos de ellas destinadas a especificar la localización de los camiones, asumiendo que pueden tener 6 valores distintos con cada una de las posiciones $D_{v_1} = D_{v_2} = \{A, B, C, D, E, F\}$. La tercera variable sería el paquete que podría tener valores entre 8 distintos $D_{v_c} = \{A, B, C, D, E, F, t_1, t_2\}$. En el siguiente cuadro 2.4 se muestra cómo quedaría codificado en SAS^+.

$$V = \{v_1, v_2, v_c\}$$

$$D_{v_1} = \{A, B, C, D, E, F\}$$

$$D_{v_2} = \{A, B, C, D, E, F\}$$

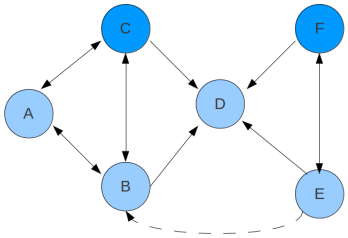
$$D_{v_c} = \{A, B, C, D, E, F, t_1, t_2\}$$

$$O = \{(\{v_1 \mapsto A\}, \{v_1 \mapsto B\}), (\{v_1 \mapsto B, v_c \mapsto B\}, \{v_c \mapsto t_1\}), ...\}$$

$$s_0 = \{v_1 \mapsto C, v_2 \mapsto F, v_c \mapsto E\}$$

$$S_* = \{v_c \mapsto B\}$$

Table 2.4: Codificación problema SAS^+



Tenga en cuenta que la ubicación D tiene cuatro entradas, pero no hay caminos de salida. El objetivo es mover un paquete situado en E a B (indicado por la flecha discontinua). Existen dos camiones inicialmente en C y F se puede utilizar para lograr este objetivo.

Figura 2.8: Problema irresoluble de *logistics*

2.8.1. Definición SAS^+ en tarea de planificación

Una tarea de planificación SAS^+ está formada por una tupla de cuatro de elementos:

- $\pi = \{V, O, S_0, S_*\}$
- Siendo los componentes que forman esta fórmula:
- $V = (v_1, v_2, ..., v_n)$ Se trata de los estados de las variables asociadas a un dominio con valores finitos Dv.
- O, define los operadores, cada operador tiene dos componentes (pre, eff), la primera parte del operador es la precondición y la segunda parte es el efecto del mismo.
- s_0, es el estado inicial.
- s_*, es el objetivo.

Un operador definido como (pre, eff) es aplicable siempre que exista un estado s que $s(v) \supset pre(v)$, siempre que la precondición $(pre(v))$ este definida. Para aplicar una acción, las precondiciones tienen que cumplirse y una vez aplicado este, el valor de v pasa a ser $eff(v)$. Esta norma quiere decir esencialmente que una variable de estado no puede estar indefinida y que para realizar cualquier acción tiene que tener el valor de la precondición. Así mismo el valor que alcanzará al ejecutar dicha acción tiene que estar entre los valores posibles que puede alcanzar dicha variable de estado. Si nos fijamos en el cuadro 2.4, que describe un problema SAS^+ el primer operador$(\{v_1 \mapsto A\}, \{v_1 \mapsto B\})$ tiene como precondición tener el valor A, y como postcondición que la misma variable v_1 tome el valor B. Si la variable estado $v1$ tiene el valor A para la variable v_1, entonces puede ejecutar la acción y pasaría a valer B (ambos valores están definidos en el ámbito de la variable estado D_{v_1}).

2.8.2. Subtarea SAS^+.

La tarea está definida por: $\pi = \{V, O, S_0, S_*\}$, que se ha visto en el apartado anterior. Entonces si existe una subtarea V' tal que $V' \subseteq V$ que sea un subconjunto de las variables de estado del problema inicial (V), entonces esta subtarea

está compuesta por una tupla de cuatro elementos $\pi' = \{V', O|V', S_0|V', S_*|V'\}$, donde $O|V' = \{(pre|V', eff|V') \,|\, (pre, eff) \in O \bigwedge eff|V' \neq 0\}$.

Se puede ver que las subtareas SAS^+ también son tareas SAS^+, con lo que si π tiene solución, entonces π' también la tendrá, aunque no sucede lo mismo a la inversa. Esta propiedad acarrea la existencia de un método para la detección de los callejones sin salida en el espacio de búsqueda de una planificación de tareas. Los callejones sin salida se producen cuando dos o más subtareas han conseguido una solución, pero la unión de ambas no se puede dar en el problema inicial. Por eso no se puede dar la implicación inversa de que si existe solución de π' exista de π .

Sin embargo, este método está limitado, en la figura 2.8 (correspondiente al cuadro 2.4 en codificación SAS^+), todas las subtareas excepto la tarea original se pueden resolver. Por ejemplo considere la subtarea $\{v_1, v_c\}$. Se puede resolver en dos etapas, en primer lugar mover el elemento de carga en el camión y la segunda mover el camión a la meta (posición B). Tenga en cuenta el hecho de que la ausencia de la variable de estado para el segundo camión de la subtarea conduce a una situación que no se puede conseguir.

Esto simplifica la planificación de tareas, evitando que desaparezcan precondiciones importantes en el camino. Para formalizar lo anterior, se introduce un nuevo concepto: el grafo causal (causal graph). Esta idea no es nueva, se ha tratado en algunos artículos de problemas jerarquícos por Knoblock y por Bacchus y Yang ambos en 1994. Más recientemente se ha aplicado en la planificación unaria en STRIPS [DB02].

2.9. Planificación heurística basada en *Causal Graph*

En primer lugar, hay que explicar la base de la heurística de este problema. Se trata de una heurística basada en *Causal Graph* [BF97]. Para eso partimos de la base del documento expuesto por Malte Helmet [HN01, Hof03] donde se explica lo siguiente.

En los últimos años, el método más representativo de planificación heurística es el algoritmo FF [HN01] que conseguía unos buenos resultados en problemas de planificación. Sin embargo, la planificación en algoritmos como el FF y HSP [BG98] todavía es algo pobre. Esto es debido a que su método de relajación ignora las "listas borradas" de los operadores en STRIPS y de esta manera se pierde información demasiado valiosa.

Por tanto esta heurística trata de trasladar problemas STRIPS a planificación formal con estados mutivaluados en una estructura causal subyacente.

La primera planificación independiente del dominio realizada de manera exitosa es HSP. Otros investigadores adoptaron el concepto general desarrollado en HSP aplicando varias modificaciones a la misma. Esto ha dado lugar a planificadores como GRT [RV01], que utiliza la heurística de la búsqueda hacia adelante entre las estrategias de planificación.

Esta heurística trata de ver cómo los conceptos de alto nivel y sus interacciones pueden ser usadas y encontradas automáticamente en sistemas de planificación independientes del dominio.

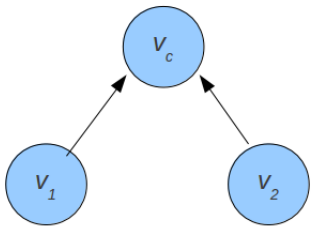
2.9.1. Definición de *Causal Graph*

Sea $\pi = \{V, O, S_0, S_*\}$ en una tarea SAS^+ , como se ha visto anteriormente. El *causal graph* $CG(\pi)$ es un diagrama (V, A) que contiene un conjunto de arcos (u, v) siendo $u \neq v$, además de existir un operador $(pre, eff) \in O$ tal que el $eff(v)$ se encuentre definido y la $pre(u)$ o $eff(u)$ también lo estén.

Esto quiere decir que el grafo contiene los arcos de la/s variable/s origen a la/s variables/s destino si el valor de la variable destino depende de la variable origen. Teniendo en cuenta que se ha debido incluir un arco en el CG, aunque esta dependencia es sólo la forma de un efecto sobre la variable de origen.

Con esta definición, un estado s puede transformarse en un estado s' si y sólo si esto es posible en la tarea inducida por los antepasados del CG de todas las variables en que difieren s y s'. Esta importante propiedad no se llevaría a cabo si los arcos entre las variables afectadas por el mismo operador no formaban parte del CG.

Esta propiedad de separabilidad en gran medida facilita la resolución de tareas donde las variables tienen pocos antepasados en el grafo. Este es el motivo por el que se han estudiado las tareas de planificación con *causal graph* acíclicos [DB02]. Un *causal graph* acíclico implica que todos los operadores son instanciados, porque los operadores con K *efectos* introducen k *ciclos* en el grafo.



Este grafo representa que la variable v_c que representa el paquete, depende de las variables v_1 y v_2 que representan los camiones

Figura 2.9: *Causal Graph* de la figura 2.8

Las tareas STRIPS con *causal graph* acíclicos son raras, pero este hecho no es necesariamente cierto en SAS^+. Por ejemplo, el dominio *logistics* entra dentro de esa clase; sin embargo, otros problemas más generales de transporte que tengan en cuenta la capacidad o el combustible pueden no ser acíclicos. El *causal graph* de la figura 2.9 sí lo es.

2.9.2. Definición de SAS^+-1

La tarea π de SAS^+-1 se trata de una tarea SAS^+ con una variable v designada de tal manera que $CG(\pi)$ tiene un arco de todas las otras variables a v y ningún otro arco. Esa variable v es llamada *variable de alto nivel*, mientras que todos los estados de esa variable son llamados *variables de bajo nivel*. El objetivo del problema debe estar definido con variables de alto nivel, y no con las variables de bajo nivel.

Todas las variables bajo nivel se pueden cambiar de forma independiente, utilizando los operadores que no tengan precondiciones de otras variables. La subtarea está compuesta por un conjunto de variables más reducido, lo que conlleva que pueda ser fácilmente resuelto por las técnicas básicas de búsqueda en grafos.

La planificación $SAS^+ - 1$ es interesante porque tiene una estructura muy simple y aún así lo suficientemente expresiva como para capturar las complejas interacciones entre variables de estado. Para mostrar esta planificación se tiene que introducir el concepto de *domain transition graph* (DTG). El DTG de una variable de estado es una representación de las formas en que la variable puede cambiar su valor, y de las condiciones que deben cumplirse para que se cambie de valor.

2.9.3. Definición DTG

Considerando una tarea en SAS^+ como el conjunto de variables V, y siendo $v \in V$. El grafo G_v es un diagrama etiquetado con vértices que contienen un conjunto de arcos (d, d'). Si existe un operador $\langle pre, eff \rangle$ donde $pre(v) = d$ o $pre(v)$ no se encuentra definido y además el $eff(v) = d'$. Este arco se etiqueta con $pre \mid (V \setminus \{v\})$.

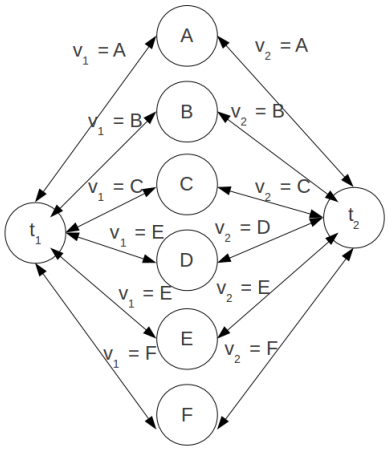
Por cada arco (d, d') con una etiqueta L, decimos que hay una transición de v de d a d' con la condición de L.

Múltiples transiciones entre los mismos valores usando diferentes condiciones son posibles. Un ejemplo de este *domain transitision graph* aparece en la figura 2.10.

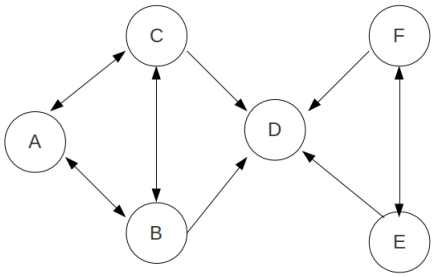
Si todos los operadores de la tarea son instanciados, existe una fuerte correspondencia entre este tipo de grafo, los estados y las transiciones de estado. Un estado se caracteriza por los valores de las variables de estado que a su vez se corresponden con los vértices en sus respectivos DTG. Para un determinado estado s, que llamamos $s(v)$ el vértice activo de G_v. La solicitud del operador en una tarea unaria SAS^+ de cambiar el valor del estado de una variable, se realiza cambiando el valor por alguno de los sucesores en el *domain transition graph*.

El plan de ejecución puede ser visto como un recorrido simultáneo de *domain transition graph*, donde la transición de un determinado grafo puede permitirse o no dependiendo de los vértices activos en otros *domain transition graph*.

La solicitud del operador instanciado en una tarea SAS^+ cambia el valor de una variable de estado, cambiando el vértice activo de la DTG por el de algún sucesor. Esta transición se permite si y sólo si sus condiciones se cumplen (restricciones en los vértices activo de otro DTG) [DD01].



(a) *Domain transition graph* de v_c



(b) *Domain transition graph* de v_1 y v_2

En la sub-figura a: Se muestra las transiciones que pueden ocurrir con la variable v_c. En sus arcos se indica el valor que tiene que tomar las otras variables para que se produzca esa transición.
En la sub-figura b: Se muestra las transiciones que pueden ocurrir con las variables v_1 y v_2, ambas variables poseen el mismo grafo. En este caso no poseen valor sus arcos porque no dependen de otras variables como se ha visto en el *causal graph*.

Figura 2.10: *Domain transition graph*

2.9.4. Algoritmo de planificación de SAS^+-1

En esta sección se describe un algoritmo polinomial para resolver las tareas SAS^+-1 . Teniendo en cuenta la tupla (V, O, s_0, s_*) que forma una tarea $SAS^+ - 1$. Tenemos que v_H se trata de una variable de alto nivel del dominio D_H.

Este algoritmo está basado en el algoritmo de Dijkstra [Knu77][3]. Consiste en encontrar un plan para D_H, no sólo con el valor deseado $s_*(v_H)$. Como en el algoritmo de Dijkstra, este encuentra el plan más corto para cualquier

[3] Se trata de un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista. Su nombre se refiere a Edsger Dijkstra, quien lo describió por primera vez en 1959.

variable de alto nivel. Esto se repite hasta no encontrar más valores alcanzables o tener un plan hasta el objetivo. Este algoritmo no es completo porque no calcula todos los planes posibles para lograr el mejor valor de las variables de alto nivel, aunque esto podría hacer que llevasen a callejones sin salida por las modificaciones de las variables de bajo nivel.

Las variables de bajo nivel son los "recursos" que son usados para lograr el objetivo de las variables de alto nivel. Este algoritmo trata de resolver el valor de cada variable de alto nivel tan rápido como sea posible, sin intentar preservar el valor de los recursos. Ya que se compromete a conseguir un plan determinado para poder alcanzar valores a todas las variables de alto nivel. El algoritmo [Hel04] es el siguiente 2.1[4] :

Algoritmo 2.1 Algoritmo SAS^+

1. Inicialización
   - $plan(d_H)$=$\begin{cases} <> & si & & s= & s_0(v_H) \\ sin & definir & todo & lo & demás \end{cases}$
   - $Cola = D_H$
2. Sea d_H un elemento de la Cola que minimiza $||plan(d_H)||$ (planes sin definir cuestan ∞). Eliminar d_H de la $Cola$. Sea $\pi = plan(d_H)$ y s el estado que resulta de aplicar π en el estado inicial. Esto se aplica a todas las transiciones de las variables de alto nivel op originarias de d_H con objetivo d'_H y precondición pre:
   - Comprobar si es posible satisfacer todas las condiciones en pre a partir de un estado s. Si no, parar y considerar la siguiente transición. Si satisface las condiciones, entonces π_L es el plan que tiene el mínimo coste que cumple todas las precondiciones. Este plan es computado por una búsqueda con el algoritmo de Dijkstra en los *domain transition graphs* de las variables de bajo nivel. Definiendo que π' es una concatenación de π, π_L y (op).
   - Si $||plan(d'_H)|| > ||\pi'||$, entonces $plan(d'_H) = \pi'$
3. Repetir el paso 2 hasta que $Cola$ no contenga ningún valor del plan definido, o el estado meta $s_*(v_H)$ es borrado de la cola, en ese caso $plan(s * (v_H))$es la solución.

Este algoritmo es completo si el *domain transition graph* de todas las variables de bajo nivel están fuertemente conectadas. En este caso siempre es posible satisfacer las condiciones de una transición de alto nivel.

2.10. Distancia entre grafos

En primer lugar se va ha repasar el concepto de grafo [W+01]. Un grafo es un conjunto de objetos llamados vértices o nodos unidos por enlaces llama-

[4] La notación $\|\Pi\|$ denota el coste del plan.

dos aristas o arcos, que permiten representar relaciones entre elementos de un conjunto.

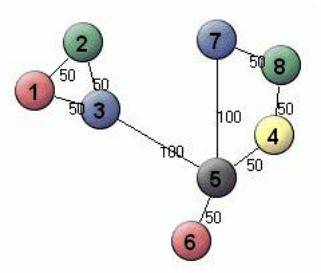


Figura 2.11: Grafo ponderado

- Los vértices o nodos en la ilustración son los círculos de colores.
- Las aristas o arcos son las líneas que conectan los vértices.
- Los pesos son los valores que aparecen encima de las aristas.

Además pueden existir dos tipos de grafos, los dirigidos y los no dirigidos. El ejemplo de la figura 2.11 es un *grafo no dirigido*. El *grafo dirigido* es aquel en el que la navegación solo se puede realizar de un vértice a otro, no indistintamente como en el anterior. Para que esto ocurriese tendría que existir dos aristas distintas con la fecha que posee el arco apuntando a su nodo correspondiente. Un ejemplo de grafo dirigido aparece en la figura 2.12.

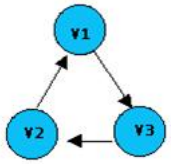


Figura 2.12: Grafo dirigido

Los grafos que se utilizan en el presente proyecto son dos y las características de los mismos son:

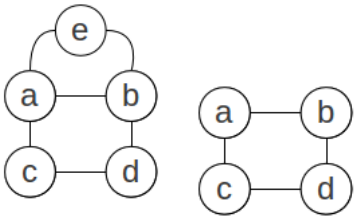
1. CG: Es un grafo dirigido, con pesos numéricos.
2. DTG: Es un grafo dirigido, con etiquetas textuales por pesos.

Debido a que poseen naturaleza distinta, se detallan dos formas distintas para conseguir comparar dichos grafos.

2.10.1. Métodos de comparación de grafos

La similitud de los grafos es considerada con la mínima distancia al cuadrado entre sus correspondientes valores. Los vértices que tienen que compararse son

establecidos por isomorfismo[5]. Teniendo en cuenta que si el grafo subisormorfo es del mismo tamaño que uno de los comparados, significa que uno uno de los grafos está contenido en el otro como se puede ver en la figura 2.13.



(a) Grafo A (b) Grafo B y grafo subisomorfo

Figura 2.13: Subisomorfismo

El principal problema de saber si dos grafos son isomorfos es que partimos de un problema que tiene una complejidad elevada, lo que conllevaría un gasto enorme de tiempo. La forma más intuitiva de conseguir que dos grafos sean isomorfos es o bien eliminando vértices o eliminando transiciones del grafo más grande. Sin embargo, simplemente con ello no se conseguiría saber si los grafos son isomorfos.

Para ello tenemos el concepto de máximo grafo común [BS98], pero este concepto también posee una complejidad elevada. Este problema consiste en permutar los valores del grafo hasta encontrar el isomorfo, siendo los dos grafos del mismo tamaño, o si uno es más grande que otro, reduciendo como anteriormente se ha comentado.

Existen algunos algoritmos de aproximación entre estructuras similares [Kam06]. Uno de ellos son las listas de distancia: Consiste en agrupar los pesos en función de sus vértices, creando una lista denominada "*dirtlist*", de esta lista se seleccionan los pesos que mejor resultado se obtengan aplicando la siguiente formula.

$$\min_{1 \leq j_1 < \cdots < j_{n-1} < m-1} \sum_{i=1}^{n-1} (x_i - y_{j_i})^2 \tag{2.1}$$

Donde esta fórmula es el sumatorio de las distancias al cuadrado del grafo bipartito que se muestra en la figura 2.14. Siendo los x_i los nodos del grafo 1, los y_j los nodos del grafo 2, y $(x_i - y_{j_i})$ la distancia entre cada uno de los vértices de los dos grafos.

[5]Isomorfismo entre dos grafos G y H es una biyección f entre los conjuntos de sus vértices $f : V(G) \to V(H)$ que preserva la relación de adyacencia. `http://es.wikipedia.org/wiki/Isomorfismo_de_grafos`

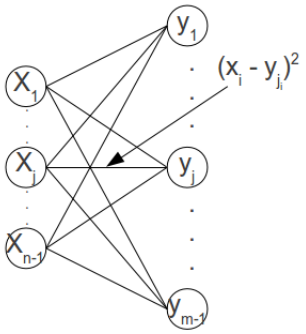


Figura 2.14: Grafo de comparación de distancias

Se seleccionan tantos números de la *distlist* como pesos tenga el grafo más pequeño de los dos. Con estos resultados se crea un grafo bipartito[6] uniendo los vértices con los pesos que se han conseguido. Para conseguir el grafo isomorfo tan solo suman el peso total de llegar desde un vértice a otro con el grafo resultante y los que obtengan menor camino son los seleccionados. Como se puede ver en la figura 2.14 se muestra el grafo bipartito de la *dirtlist* y el valor del grafo isomorfo se consigue con la aplicación de la fórmula vista.

Otro sistema de comparación de grafos se trata de la distancia métrica basada en el máximo común subgrafo. Como su propio nombre indica se puede percibir que parte del concepto es similar a la forma anteriormente descrita. Y así es, en este sistema también necesitamos conocer si existe un grafo isomorfo que se corresponda con el grafo que estamos comparando. Sin embargo, en esta técnica no se utilizan los pesos del grafo, sino la cantidad de vértices que tiene. Esto verdaderamente resultará útil con el grafo *domain transition graph*, ya que las etiquetas (pesos) de las aristas son textuales. La fórmula en la que se basa este método de comparación es la siguiente:

$$d(G1, G2) = 1{,}0 - (|mcs(G1, G2)|)/max(|G1|, |G2|)$$

- Siendo mcs: el máximo común subgrafo.
- |G1| el número de vértices del grafo 1.
- |G2|el número de vértices del grafo 2.

2.11. Métodos de análisis de datos

La minería de datos (DM, *Data Mining*) consiste en la extracción no trivial de información que reside de manera implícita en los datos. Dicha información era previamente desconocida y podrá resultar útil para algún proceso. En otras palabras, la minería de datos prepara, sondea y explora los datos para sacar la información oculta en ellos.

[6]Un Grafo bipartito se denomina en Teoría de grafos a un grafo cuyos vértices se pueden separar en dos conjuntos disjuntos$V1$ y $V2$ y las aristas siempre unen vértices de un conjunto con vértices de otro: `http://es.wikipedia.org/wiki/Grafo_bipartito`

Bajo el nombre de minería de datos se engloba todo un conjunto de técnicas encaminadas a la extracción de conocimiento procesable, implícito en las bases de datos. Está fuertemente ligado con la supervisión de procesos industriales ya que resulta muy útil para aprovechar los datos almacenados en las bases de datos.

Las bases de la minería de datos se encuentran en la inteligencia artificial y en el análisis estadístico. Mediante los modelos extraídos utilizando técnicas de minería de datos se aborda la solución a problemas de predicción, clasificación y segmentación.

En este apartado se explican brevemente las técnicas de análisis de datos. Las técnicas de minería de datos se clasifican en dos grandes grupos: supervisados o predictivas y no supervisadas o descriptivas.

Las predicciones se utilizan para prever el comportamiento futuro de algún tipo de entidad mientras que la descripción para ayudar a su comprensión. De hecho, los modelos predictivos pueden ser descriptivos y los modelos descriptivos pueden emplearse para realizar predicciones. De esta forma, hay algoritmos o técnicas que pueden servir para distintos propósitos. Por ejemplo, las redes de neuronas pueden servir para predicción, clasificación e incluso aprendizaje no supervisado.

2.11.1. Aprendizaje no supervisado

El aprendizaje inductivo no supervisado estudia sin ayuda del maestro; es decir, se aborda el aprendizaje sin supervisión, que trata de ordenar los ejemplos en una jerarquía según las regularidades en la distribución de los pares atributo - valor sin la guía del atributo especial clase. Éste es el proceder de los sistemas que realizan clustering conceptual y de los que se dice también que adquieren nuevos conceptos.

Dentro del aprendizaje no subervisado la técnica más importante es el *Clustering*. Esta técnica permite la identificación de tipologías o grupos donde los elementos guardan gran similitud entre sí y muchas diferencias con los otros grupos.

- **Clustering númerico**. Uno de los algoritmos más utilizados para hacer clustering es el k-medias (*k-means*) [M$^+$67], que se caracteriza por su sencillez. En primer lugar se especifican los clusters que se van a crear, para lo cual se seleccionan k elementos aleatoriamente, que representan el centro o la media del cluster. A continuación el resto de datos es asignado a un centro usando la distancia Euclídea. Así se calcula un nuevo centroide con los datos de cada cluster, este proceso se repite hasta que se asignan los mismos ejemplos a los mismos clusters ya que el punto central se ha estabilizado.

- **Clustering conceptual**. El algoritmo k-medias se encuentra con un problema cuando los atributos no son numéricos, ya que en ese caso la distancia entre ejemplares no está tan clara. Para resolver el problema Michalski [MS83] presenta la noción de clustering conceptual, basado en la vecindad entre los elementos de la población. Una de las principales motivaciones de la categorización de un conjunto de ejemplos, es la predicción de características de las categorías que heredan subcategorías. Esta conjetu-

ra es la base de COBWEB [Fis87]. El objetivo de COBWEB es hallar un conjunto de clases o clusters que maximice la utilidad de la categoría.

- **Clustering probabilístico**. Los algoritmos de clustering anteriores presentan ciertos defectos entre los que destacan la dependencia que tiene el resultado del orden de los ejemplos y la tendencia de estos algoritmos al sobre ajuste. Por lo que la base de este tipo de clustering se encuentra en un modelo estadístico llamado mezcla de distribuciones. Cada distribución representa la probabilidad de que un objeto tenga un conjunto particular de pares atributo valor, si se supiera que es miembro de ese cluster. Un ejemplo es el algoritmo EM, que empieza adivinando los parámetros de las distribuciones y, a continuación, los utiliza para calcular las probabilidades de que cada objeto pertenezca a un cluster y usa esas probabilidades para reestimar los parámetros de las probabilidades, hasta converger.

2.11.2. Aprendizaje Supervisado

En el aprendizaje inductivo supervisado existe un atributo especial, normalmente denominado clase, presente en todos los ejemplos que especifica si el ejemplo pertenece o no a un cierto concepto, que será el objetivo del aprendizaje. Dentro de este tipo de aprendizaje se pueden distinguir dos tipos de técnicas: la predicción y la clasificación [WK91].

2.11.2.1. Predicción

Es el proceso que intenta determinar los valores de una o varias variables, a partir de un conjunto de datos. La predicción de valores continuos puede realizarse por técnicas de regresión [SM96, WMMY02].

- **Regresión lineal**: Similar al modelo anterior; este modelo se expresa como: $Y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon$ [MPVV01]

- **Regresión no lineal**. En muchas ocasiones no se muestran una dependencia lineal. Esto es lo que sucede si, por ejemplo, la variable respuesta depende de las variables independientes según una función polinómica, dando lugar a una regresión polinómica.

- **Árboles de predicción**. Estos árboles de decisión almacenan en cada nodo hoja un valor de clase que consiste en la media de todas las instancias que se clasifican con esa hoja, en cuyo caso se habla de un árbol de regresión, también denominado árbol de modelos; este es el caso del algoritmo M5 [WFT$^+$99].

- **Reglas de predicción**. Son un conjunto de reglas que definen la clase. Estas reglas pueden ser las denominadas *"si-entonces"* y además pueden tratarse de rectas de regresión, como es el caso de M5Rules. M5Rules, que se trata de un clasificador encargado de crear una serie de decisiones de los problemas de regresión por separado utilizando divide y vencerás (*separate-and-conquer*). En cada iteración se construye un modelo de árbol y hace que el "mejor" de la hoja en una regla [HHP99, WW96].

2.11.2.2. Clasificación

La clasificación es el proceso de dividir un conjunto de datos en grupos mutuamente excluyentes [WK91, Hel04] , de tal forma que cada miembro de grupo esté lo más cerca posible de otros y grupos diferentes estén lo más lejos posible de otros, donde la distancia se mide con respecto a las variables especificadas, que se pueden predecir.

- **Tabla de decisión**. Constituye la forma más simple y rudimentaria de representar la salida de un algoritmo de aprendizaje, que es justamente representado como la entrada. Estos algoritmos consisten en seleccionar subconjuntos de atributos y calcular su precisión para predecir o clasificar los ejemplos.
- **Árboles de decisión**. Pueden interpretarse esencialmente como una serie de reglas compactadas para su representación en forma de árbol. Los primeros árboles de decisión fueron diseñados por Quinlan [Qui93, Qui86], los sistemas ID3 y C4.5.
- **Reglas de clasificación**. Las técnicas de inducción de reglas [Qui87, Qui93] permiten la generación y contrate de árboles de decisión, o reglas y patrones a partir de los datos de entrada. Los ejemplos más significativos son el algoritmo 1R [Hol93], PRISM [Cen87], PART [Qui93].
- **Clasificación Bayesiana** [DH73]. Se trata de clasificadores estadísticos, que pueden predecir tanto las probabilidades del número de miembros de clase, cómo la probabilidad de que una muestra dada pertenezca a una clase particular.
- **Aprendizaje basado en ejemplares o instancias** [BC96]. Tiene como principio de funcionamiento, en sus múltiples variantes, el almacenamiento de ejemplos: en unos casos todos los ejemplos de entrenamiento, en otros solo los más representativos.
- **Algoritmo K vecinos (KNN)**. Es un método de clasificación supervisada (Aprendizaje, estimación basada en un conjunto de entrenamiento y prototipos) que sirve para estimar la función de densidad $F(x/C_j)$ de las predictoras x por cada clase C_j [AKA91].
- **Redes de neuronas.** Constituyen una técnica inspirada en los trabajos de investigación, iniciados en 1930, que pretendían modelar computacionalmente el aprendizaje humano llevado a cabo a través de neuronas [RMDUoCS86, CR95], un ejemplo son el perceptrón simple y el perceptrón multicapa.
- **Lógica borrosa.** Surge de la necesidad de modelar la realidad de una forma más exacta evitando precisamente el determinismos o la exactitud [Zad65]. En otras palabras, lo que la lógica borrosa permite es el tratamiento probabilístico de la categorización de un colectivo [Zad65].
- **Técnicas genéticas**. Estas técnicas tienen su inspiración en la biología con las redes de neuronas [Gol89, MM92, Fog94]. Estos algoritmos representan el modelo matemático de cómo los cromosomas en un marco evolucionista alcanzan la estructura y composición más óptima para la supervivencia.

Capítulo 3

Desarrollo del proyecto

En este capítulo se presenta una descripción general del proyecto y las distintas fases en las que se ha compuesto el proyecto.

3.1. Descripción general del sistema

El sistema se compone de tres partes principales. La primera parte se trata de la obtención y procesamiento de los problemas que se ha realizado a partir del planificador LAMA. Como se muestra en la figura 3.1, se parte de unos ficheros que utiliza LAMA de manera auxiliar para almacenar los problemas en formato SAS^+. A partir de esos ficheros y el plan propuesto se transforman a un formato independiente (Ficheros pos-procesado), que contienen los DTG, CG. Cuando se han reunido un conjunto de datos (mínimo dos problemas) se pasa a la segunda parte del sistema, la comparación de los grafos. Esta comparación consiste en la aplicación de una serie de algoritmos para obtener una medida de similitud entre los problemas pasados.

Después de tener un conjunto de resultados de comparación (fichero de datos pos-comparación), se inicia la tercera parte del sistema, el análisis de los datos para la extracción de características de los problemas. Para ello se divide los ficheros de entrada en función de sus dominio y sus funciones objetivo. Cuando todos estos ficheros han sido creados se pasan por el programa Weka, para la realización de modelos. Cuando se han construido los modelos correspondientes, se analizan para saber si las medidas de distancia de los grafos son útiles. Además de ver que características son las más relevantes para los problemas de planificación.

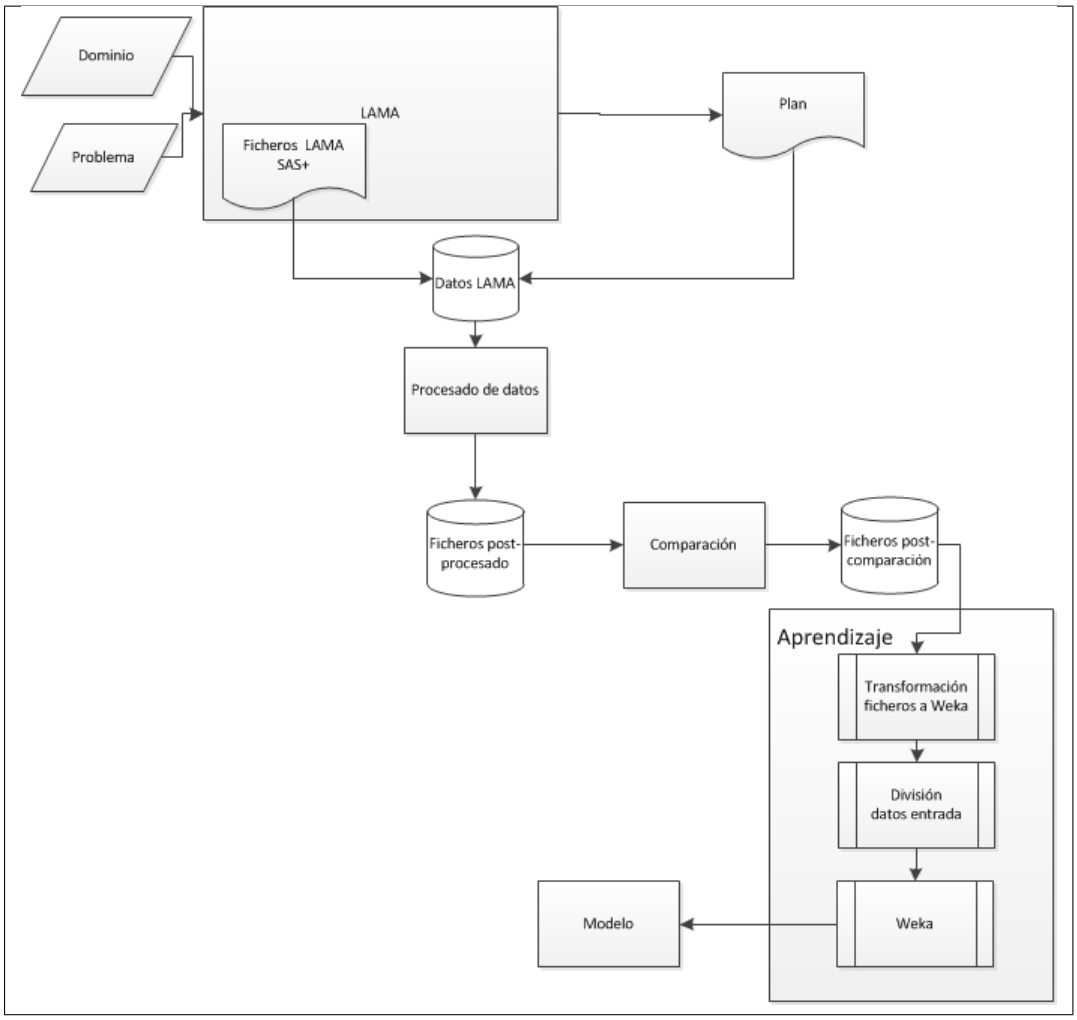


Figura 3.1: Esquema general del sistema

3.2. Obtención de los ficheros post-procesado y post-comparación.

Para la obtención de los datos se ha utilizado el programa LAMA (anteriormente explicado en el apartado 2.3.6). Sin embargo, no se han usado los propios ficheros que devuelve el programa, sino que se han creado unos propios. Estos ficheros son más sencillos que los que se ofrecen y cualquier usuario puede entenderlos más rápidamente. Además, con esto hacemos que la entrada sea independiente del planificador. Hay que tener en cuenta que no se devuelven los mismos valores que se devuelven en LAMA, sino que están expresados según la codificación de los problemas en SAS^+, salvo en el caso de los operadores, los cuales poseen una leve simplificación. Por otra parte, debido a que todas las estructuras de datos que proporcionaba LAMA se encontraban codificadas con valores numéricos, estos se han tenido que decodificar para que ofrecieran los

valores reales de problema.

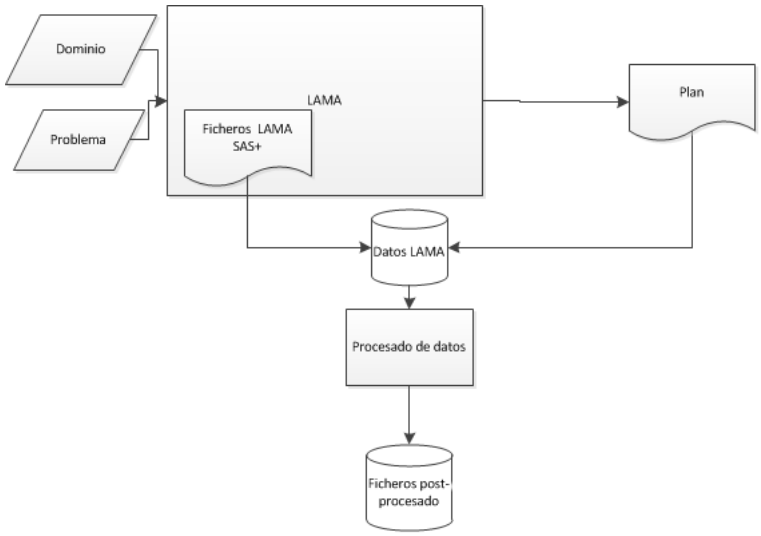


Figura 3.2: Obtención ficheros de datos

En primer lugar, se han cargado los valores de las variables sin ordenar y se han colocado en función de la importancia; es decir, primero las variables que por sí solas no alcanzan el objetivo, variables de bajo nivel, y en último lugar las que alcanzan el objetivo, variables de alto nivel. Este sistema es propio de LAMA y se utiliza para que el orden relativo de las variables codificadas sea similar en todos los problemas del mismo dominio. A continuación se pasa a leer el estado inicial, en el que se encuentran las variables del sistema. Después, se pasa a leer una de las estructuras fundamentales de este sistema: los *domain transition graph* del problema. Al igual que ocurre con los estados iniciales, hay que iterar sobre la lista de valores para conseguir los valores de las variables. Sin embargo, en este caso el valor de las etiquetas que tienen las transiciones de estos grafos no se consigue de manera tan trivial. En el grafo original, la etiqueta del grafo tiene un valor numérico, siendo este valor el correspondiente con una de las operaciones instanciadas que ha creado el sistema. No obstante, no se utilizará el valor completo del operador instanciado, sino, simplemente, el nombre original de la operación del problema. Esto se debe a que se quiere "abstraer" los grafos, es decir, que se puedan comparar dos grafos a través de sus etiquetas. Si dejásemos el valor completo, nunca se podrían encontrar valores similares. Esto ocurriría porque tendrían que ser iguales cada uno de los parámetros de este operador, cuando lo que interesa es ver que la acción sea la misma, aunque los parámetros sean distintos.

La última estructura que se introduce es el *causal graph*. Este grafo no sufre apenas modificación, ya que se trata de la relación entre las variables, no de sus valores (como en el caso anterior). Para conseguir esto, simplemente se añade "*var*" al número de variable, para poder diferenciar el peso de las mismas.

En lo que se refiere al motor de búsqueda de LAMA, nuestro fichero guarda cada una de las soluciones encontradas. Estas soluciones, que fueron encontradas con la ayuda de distintas heurísticas, se utilizan para evaluar los resultados

ofrecidos por el sistema.

Una muestra de cómo está representado el problema es la que aparece en el cuadro. 3.1. El fichero de entrada completo se encuentra en el apéndice A sección A.1.

problem | logistics
DTG
<num_var> <numero_estado_1> ...<numero_estado_n>
<estado inicial 1>
...
<estado inicial n>
var: <1> <estado1>+<estado2>+<nombre transición>
...
var: <n> <estadon>+<estadok>+<nombre transición>
CG
var: <n> - var: <m>, <peso1> [- var: <z>, <peso2>]
H(i): <valor 1>
[...
H(i): <valor n>]
result: 1
[<linea de solución 1>
...
<linea de solución n>]
f
Plan length: <numero pasos> step(s).
Expanded <numero estados 1> state(s).
Generated <numero estados 2> state(s).
Search time: <tiempo> seconds
Total time: <tiempo total> seconds
Objetos: <numero objetos>
Literales: <numero literales>
Metas: <numero metas>
end

Cuadro 3.1: Formato de entrada de problema con solución

Este fichero se compone en tres partes fundamentales, la primera de ellas se encuentra comprendida desde el token *problem*|*logistics*, hasta antes del token *CG*.

En la primera línea se puede distinguir entre un problema *logistics* o cualquier otro tipo de problema. Esta división se debe a que en el dominio *logistics* es posible conseguir "tipos de sus variables" y en el resto de los dominios no. En la siguiente línea aparece el token DTG, que indica que empieza todo el conjunto de grafos *domain transition graph*. En la siguiente línea, aparece en primer lugar el número de variables $< num - var >$, y separado por espacios los valores iniciales que tienen cada una de ellas $< numero - estado - 1 > ... < numero - estado - n >$. Hasta que aparezca en una línea el token *var* :, cada línea corresponde al estado inicial en forma textual, el numero de líneas tiene que coincidir con el numero de variables $< num - var >$. En las líneas siguientes $var :< 1 >< estado1 > + < estado2 > + < nombretransición >$, correspon-

den a cada transición del conjunto total de los grafos. Siendo $< estado1 >$ el estado desde el que transita, $< estado2 >$ el estado que va a transitar y $< nombretransicion >$ el nombre de la acción por la que transita, es decir, el nombre del arco. Para saber a que grafo pertenecen, solo hay que tener en cuenta el número de variable. Este número es justo el que se encuentra después del token *var* :. En la última parte de este apartado aparecen los valores de las heuristicas hasta conseguir el plan. Estos valores se inician con $H(i)$: y a continuación el valor de la heurística, aunque aparezcan todos sólo se utiliza la heurística del estado inicial (el primer valor que aparece de este conjunto).

Ya en la segunda parte aparece el token *CG,* esto indica el inicio del *causal graph*. Donde cada línea simboliza una transición $var :< n > -var :< m >, < peso1 > [-var :< z >, < peso2 >]$. Siendo $var :< n >$ la variable origen de la transición, $var :< m >$ la variable destino y $< peso1 >$ el valor de la transición. Además en cada línea se incluye el resto de transiciones desde esa variable y se encuentra representado como $-var :< z >, < peso2 >$, siendo la $var :< z >$ la variable destino y $< peso2 >$, el peso de esta transición. Esto puede ocurrir hasta n veces. Dentro de esta parte también se encuentra los valores conseguidos en la heurística. Estos valores se representan con $H(i)$: y a continuación el valor que poseen $< valorn >$. Estos valores tienen que aparecer al menos una vez.

La tercera parte del fichero forma parte de la solución y se inicia con la línea *result* : 1. Después aparece la solución de la misma. En el caso de que la solución sea no aplicar ninguna acción, no habrá ninguna línea dedicada a esta sección. Cuando ha finalizado la solución aparecerá en una línea el token f. En esta parte, aparecen el resto de datos no derivados de los grafos. En primer lugar la longitud del plan, que corresponde con el número de líneas que comprenden entre el *result* : 1 y f. En la siguiente línea los nodos evaluados denotados como *Expanded* y a continuación el numero de nodos $< numeroestados1 >$ $state(s)$. El formato de los nodos generados es igual, salvo que se cambia el token del principio a *Generated*. Después aparecen los tiempos, el primero de ellos se trata del tiempo de búsqueda denotado con *Search time* :, el numero de segundos $< tiempo >$ *seconds*; el tiempo total es igual que el anterior salvo que empieza por *Total time*. Los tres últimos datos son, el número de objetos, el número de literales y el número de metas en la descripción del problema. Donde su formato es el $nombre : < valor >$, siendo los nombres utilizados: *Objetos*, *Literales* y *Metas*.

Si queremos comparar con un problema que LAMA no ha resuelto se utiliza un formato similar, pero sin que aparezca la solución como se ve en el cuadro 3.2. El significado es el mismo que el de las dos primeras partes del fichero anterior.

Además, se incorporará de manera opcional un tercer fichero, con la solución del problema a resolver, para poder evaluar el acierto que ha tenido el sistema para encontrar la solución. Este fichero será simplemente la parte de las soluciones y se introducirá de la misma manera que se ha explicado anteriormente.

problem / logistics
DTG
<num_var> <numero_estado_1> ...<numero_estado_n>
<estado inicial 1>
...
<estado inicial n>
var: <1> <estado1>+<estado2>+<nombre transición>
...
var: <n> <estadon>+<estadok>+<nombre transición>
CG
var: <n> - var: <m>, <peso1> [- var: <z>, <peso2>]

Cuadro 3.2: Formato de problema sin la solución al final

3.3. Desarrollo del sistema

En este apartado se explican todas las fases de implementación y los algoritmos empleados en la comparación de los grafos.

3.3.1. Almacenamiento de problemas

En el diagrama UML mostrado en la Figura 3.3 se representan las principales clase usadas para poder almacenar y manipular los datos ofrecidos por los ficheros de entrada.

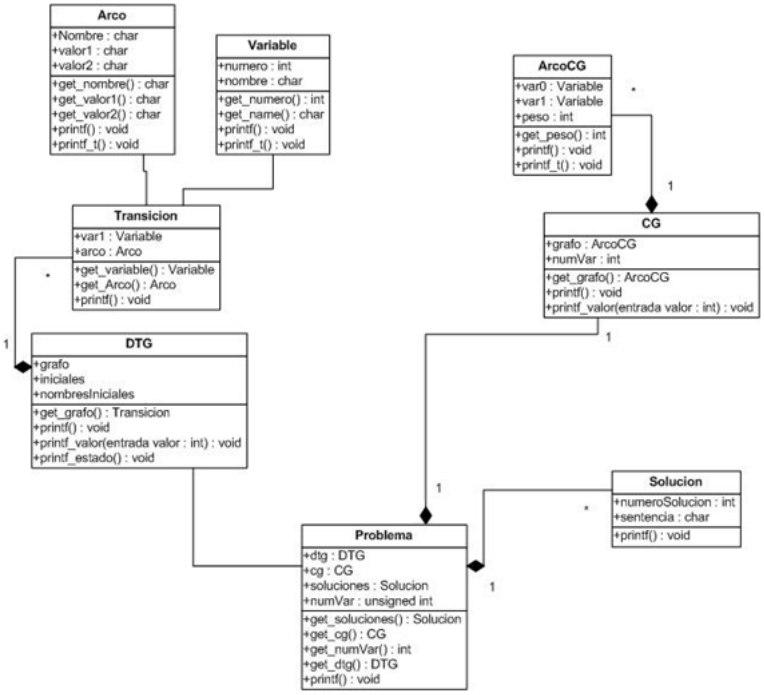


Figura 3.3: Composición del problema.

Las clases son cada uno de los elementos básicos que se encuentran en la estructura del dominio en SAS^+, formando entre todas el concepto de problema. Esta estructura se utiliza para almacenar los ficheros leídos en el apartado anterior. El conjunto de problemas resueltos forma una colección de problemas para la comparación, mientras que el problema a resolver se encuentra en un único fichero con la estructura anterior.

Como se indicó antes, esta es sólo la estructura principal del sistema, y por lo tanto, no están incluidas todas las clases, existiendo otras clases destinadas a la comparación de los grafos existentes del problema, así como a métodos de selección y evaluación de los resultados obtenidos.

3.3.2. Comparación de los *Causal Graph*

La primera subestructura que podemos encontrar en un problema es el *causal graph*. El valor de las aristas corresponde con el número de operadores que inducen esa causalidad. Los grafos que vamos a tratar en este apartado son grafos dirigidos, con un valor numérico para el peso que tienen sus aristas.

La forma más sencilla de comparar dos grafos de este tipo es mediante el uso de matrices cuadradas. De esta manera, cada valor de la matriz corresponde con el valor de un arco dirigido. Estas matrices son usadas después para comprobar si los grafos son isomorfos. Para ello se usa el siguiente algoritmo 3.1:

Algoritmo 3.1 Comparación *causal graph*

---

1. Comparación de la matriz a su totalidad.
2. Si $Tam = n$ y $FilaInicio = 0$ y $ColumaInicio = 0$: Saltar a paso 5.
3. Si no es igual (paso 2), Entonces: $Tam = n - 1$.
   a) Comprobar hasta encontrar isomorfo siendo $FilaInicio = x$, $ColumaInicio = y$, siendo $0 \leq x \leq n$ y $0 \leq y \leq n$.
   b) Si encuentran solución: $Tam = n - 1$ $FilaInicio = x$, $ColumaInicio = y$. Saltar al paso 5.
   c) Sino repetir paso b hasta que $Tam = 2$.
4. Si no se ha encontrado solución:
   a) Permutar filas. Volver al paso 3 (a).
   b) Sino Permutar columnas. Volver al paso 3 (a).
   c) Sino Permutar filas y columnas. Volver al paso 3 (a).
   d) Sino encuentra submatriz saltar al paso 6.
5. Devolver matriz.
6. Salir del algoritmo.

Este algoritmo tiene como entradas las dos matrices cuadradas de los *causal graph*. En primer lugar se comparan las dos matrices a su totalidad. Si el tamaño

de la matriz isomorfa es n ($Tam = n$) y la fila de inicio y la columna de inicio es igual a 0 ($FilaInicio = 0$ y $ColumnaInicio = 0$), es que son dos matrices exactamente iguales y se sale del algoritmo. Si no es así, se resta 1 el tamaño de la submatriz isomorfa ($Tam = n-1$) y se realiza las comprobaciones realizando desplazamientos en las filas y las columnas hasta $Tam = 2$. Si con el desplazamiento no se ha encontrado la solución, se permuta las filas, y se repite los pasos anteriores. Si con las permutaciones de filas no se encuentra la solución, se permutan las columnas y se repite los pasos anteriores. Y en ultimo lugar se permutan filas y columnas para conseguir la solución. Si en ninguno de los casos se obtiene solución se sale del algoritmo sin devolver la matriz isomorfa.

Una vez que hemos aplicado el algoritmo. Se aplica una fórmula de distancia que aparece en primer lugar en el apartado 2.10.1

$$\min_{1 \leq j_1 < \cdots < j_{n-1} < m-1} \sum_{i=1}^{n-1} (x_i - y_{j_i})^2$$

Esta fórmula consiste en el valor mínimo de dos grafos isomorfos. Se aplica a las dos matrices originales que tengan en común esos elementos; por ejemplo, si la submatriz empieza en la columna 1 fila 2 y tiene tamaño 2, la fórmula se aplica a las dos matrices dentro de ese rango. También hay que tener en cuenta que si el algoritmo no encuentra ninguna submatriz en común, está fórmula valdrá 0.

Esta medida no permite conseguir una medida de similitud muy precisa. Ya que la intención de esta medida consiste en conseguir el isomorfismo de dos grafos. Y lo que queremos conseguir es una distancia entre los dos grafos y para ello hay que introducir una medida de desorden. Además, se podría caer en el error de alcanzar muchos valores similares, ya que sólo se compara una parte de las matrices completas. Por lo tanto, se introdujo una medida de desorden. Esta medida consiste en la adición de los valores no incluidos en la submatriz. Para ello se utiliza la fórmula de distancia, teniendo en cuenta que el segundo término en este caso va a valer 0. Todo ello nos lleva a que la formula final de comparación sea de la siguiente forma:

$$\min_{1 \leq j_1 < \cdots < j_{n-1} < m-1} \sum_{i=1}^{n-1} (x_i - y_{j_i})^2 + \sum_{i=n}^{n+k_1} (x_i)^2 + \sum_{i=n}^{n+k_2} (y_{j_i})^2$$

.

Siendo los nuevos términos de la misma:

- $\sum_{i=n}^{n+k_1} (x_i)^2$: La suma del resto de los valores que no forman la submatriz isomorfa del grafo 1, en el caso de que sea toda la matriz valdrá 0.
- $\sum_{i=n}^{n+k_2} (y_{i_i})^2$: La suma del resto de los valores que no forman la submatriz isomorfa del grafo 2, en el caso de que sea toda la matriz valdrá 0.

Las propiedades de esta fórmula son:

- El límite inferior es 0 $\Rightarrow$ indica que los grafos son exactamente iguales.
- El límite superior es infinito.
- El valor que se obtiene no es escalable entre problemas de distinto tamaño.

3.3.3. Comparación de los *Domain Transition Graph*

La segunda subestructura que podemos encontrar en un problema son los DTG. Estos grafos relacionan los valores de las variables. Los grafos que vamos a tratar en este apartado son también grafos dirigidos. El valor que tienen sus aristas equivalen a las acciones que se pueden realizar en un plan. Estos grafos no son tan simples de tratar como los anteriores, porque no existe una representación numérica que represente fielmente la realidad.

En este caso, para saber si dos grafos son isomorfos, es necesario comparar los valores de las aristas. Para ello es necesario saber que existe la misma cantidad de acciones entre objetos del mismo tipo. Para su determinación se aplica el algoritmo descrito en el Cuadro 3.2.

Algoritmo 3.2 Algoritmo para la determinación de isomorfismo entre grafos.

```
Por cada problema
Por cada variable


    Se cuenta el número de acciones diferentes, con su nombre
    correspondiente
        Se cuenta la cantidad de cada tipo
        Se comparan los nombres hasta encontrar una variable
        que tenga los subvalores correspondientes
        Si se encuentra
            Se pasa a la siguiente variable
        Si no se encuentra


            No existe grafo isomorfo para esa variable
    done
    Si se ha encontrado grafo isomorfo para todas las variables
        ¡Problemas isomorfos!
    Sino
        ¡Problemas no isomorfos!
done
Fin del algoritmo
```

Este algoritmo tiene como entrada todos los grafos *domain transition graph* de los dos problemas. Para cada variable del primer problema (correspondiente con un grafo) busca en las variables del segundo problema hasta encontrar una que poseea las mismas acciones. Una vez encontrada, cuenta en cada grafo la cantidad de acciones que hay de cada tipo y las compara para devolver el número de nodos del grafo que son isomorfos. Si no encuentra un grafo que poseea las mismas acciones es que no existe grafo isomorfo para esa variable.

Una vez que se ha determinado que existe una matriz isomorfa, se comparan los valores de todos los grafos del problema. La fórmula que representa esta comparación es exactamente la misma que la resultante en el apartado anterior:

$$\underset{1 \leq j_1 < \cdots < j_{n-1} < m-1}{min} \sum_{i=1}^{n-1} (x_i - y_{j_i})^2 + \sum_{i=n}^{n+k_1} (x_i)^2 + \sum_{i=n}^{n+k_2} (y_{j_i})^2$$

De tal manera que:

- $\underset{1 \leq j_1 < \cdots < j_{n-1} < m-1}{min} \sum_{i=1}^{n-1} (x_i - y_{j_i})^2$ La resta al cuadrado de los valores de los arcos de las submatrices resultantes isomorfas.
- $\sum_{i=n}^{n+k_1} (x_i)^2$: La suma del resto de los valores que no forman la submatriz isomorfa del grafo 1, en el caso de que sea toda la matriz valdrá 0.
- $\sum_{i=n}^{n+k_2} (y_{i_i})^2$: La suma del resto de los valores que no forman la submatriz isomorfa del grafo 2, en el caso de que sea toda la matriz valdrá 0.

De esta manera se consigue alcanzar la fórmula de distancia que se aplica en el caso del CG, pero teniendo en cuenta que el valor de sus aristas puede ser 1 ó 0.

3.3.4. Comparación de los estados iniciales

La siguiente subestructura que podemos encontrar en un problema son los estados iniciales. Estos son valores que tienen cada una de las variables en el estado inicial.

Para la realización de esta comparación hay que codificar los valores de los estados a números. Estos números están relacionados con los estados que alcanzan en las transiciones en el DTG de cada variable. De tal manera, sumando una posición te mueves por un arco del grafo. Esta codificación es una codificación basada en la realidad, donde la medida de distancia puede tener un significado válido. Por lo tanto, para encontrar dicha distancia se requiere, simplemente, aplicar la siguiente formula:

$$0 \leq i \leq max(var_p1, var_p2) \sum_{i=1}^{n-1} (x_i - y_i)$$

De tal manera que:

- $0 \leq i \leq max(var_p1, var_p2) \sum_{i=1}^{n-1}$ Simboliza la comparación de 0 al número máximo de variables que exista en cualquiera de los dos problemas, o el máximo de variables en el grafo isomorfo resultante.
- $\sum_{i=1}^{n-1} (x_i - y_i)$: El sumatorio de la resta de todos los valores de los estados iniciales incluidas en el sumatorio.

Hay que tener en cuenta que los problemas pueden no ser del mismo tamaño. En ese caso, el estado del problema que tenga menos será 0.

Además, para que se aplique esta fórmula se ha tenido en cuenta que las variables que se comparan representan lo mismo en los distintos problemas, de la siguiente manera:

- Si son completamente isomorfos los grafos: Resta de todas las variables y ayuda a saber qué problema es más parecido.

- Si se encuentra un isomorfismo: se restan solo las submatrices que forman el isomorfismo. En el caso de que esta resta dé 0, significa que un grafo está contenido en el otro.

- Si no se encuentra isomorfismo: se restan todas las variables, pero no aporta información adicional a la comparación.

3.4. Análisis de los problemas de planificación

Después de la comparación de distintos problemas con los algoritmos explicados en los puntos anteriores, se cambió el formato de los datos para realizar la experimentación. El formato de estos datos se trata de uno prefijado ya que se va a realizar la experimentación con el programa *Weka.* Además de tener que cambiar el formato de los datos, también hay que cambiar la extensión del fichero a "*.arff*". Con estos nuevos ficheros se pasan una serie de pruebas para la construcción de unos modelos que nos digan que características son las relevantes en los problemas de planificación y si la medida de distancia de los grafos es relevante.

3.4.1. Transformación de los ficheros a Weka

Una vez que se han obtenido todos los resultados de comparación con el conjunto total de problemas, se ha cambiado el formato de salida del sistema expuesto en el apéndice B apartado B.2, al formato mostrado dentro de ese mismo apéndice en el apartado B.2.

Para ello, se introdujo en todos los ficheros una cabecera común (ya que todos los ficheros poseen los mismos atributos) y posteriormente en cada línea un experimento realizado, formando un total de tres ficheros con 900 líneas cada fichero. Cada uno de estos ficheros corresponde con un dominio utilizado.

3.4.2. División de los ficheros de entrada

Como se quiere comparar los resultados obtenidos en función de las variables objetivo, es necesario dividirlos en distintos ficheros. Estas divisiones se han descompuesto fundamentalmente en solo atributos de entrada básicos, atributos de entrada básicos con los relacionados con grafos y atributos de entrada básicos con atributos de entrada derivados y los relacionados con grafos. Además de estas divisiones, se ha realizado un preprocesado de datos aplicando filtros para quitar los datos fuera de rango y para realizar selección de atributos. De esta manera se puede observar que atributos son más relevantes en todos los casos.

3.4.3. Experimentos en Weka

Con todos los ficheros que se han creado en el apartado anterior, se han realizado una serie de grupos divididos por dominio y función objetivo. Para saber que conjunto de datos obtiene mejor solución y comprobar que la medida creada para la comparación de grafos es relevante para la predicción de los valores de salida.

Para ello se han utilizado una serie de técnicas como son la regresión lineal, las reglas de predicción, reglas de clasificación y aprendizaje basado en ejemplares. Con estas técnicas se quiere ver cual es el modelo más sencillo que obtiene un resultado adecuado. Y qué modelo obtiene menos errores.

3.4.4. Obtención de los modelos

Cuando se han ejecutado todos los experimentos se ha realizado una comparación para saber cuales son los mejores resultado de cada prueba y ver que algoritmos son los que mejores resultados dan. Con esto se han obtenido la relación de atributos más importantes y la relevancia que ha tenido el valor de comparación de los grafos.

Capítulo 4

Experimentación

En este capítulo se va a mostrar como se ha pasado de la comparación de los problemas de planificación al programa Weka. Se detalla el conjunto de problemas empleados en esta fase y la configuración de los mismos. Posteriormente de detallan las técnicas de análisis de datos así como los ficheros de datos que se han utilizado y los valores que se han obtenido de los experimentos.

La segunda parte de este capítulo muestra los experimentos separados por dominios y las conclusiones obtenidas de los mismos.

4.1. Obtención de los datos

En este apartado se explica cómo se pasa de la comparación de los resultados del sistema implementado a fichero Weka, para realizar el análisis.

4.1.1. Salida del programa

Con todo lo que se ha explicado en el capítulo anterior se han generado un conjunto de ficheros tal y como se muestran en el apéndice B apartado B.1. Para pasarlos a formato Weka, se ha prescindido de los resultados que aparecen en primer lugar, las matrices de los problemas y las submatrices encontradas. Con esta segunda parte de la solución se han transformado al formato que aparece en el apartado B.2. Esto se ha realizado con la permutación del orden que aparece en el primer formato. En primer lugar aparece el nombre de todas las variables. y después de añadir el token "*@data*" todos los datos que muestra el programa separados por comas. Los datos que aparecen son el número de problema, la comparación del *causal graph*, la comparación del *domain transition graph*, la comparación de los estados iniciales, la comparación del resto de los atributos, así como los valores individuales en los dos problemas. Así, cada línea del fichero se trata de la comparación de un problema con el resto, llegando a ser un total de 900 líneas cada uno de los tres ficheros que se han generado. Cada uno de los tres ficheros esta dedicado a un dominio descompuesto como aparece en el siguiente apartado.

4.1.2. Tiempo de generación de ficheros

En este apartado vamos a exponer el tiempo computacional expresado en segundos que se ha empleado en la realización de los ficheros. Como se ha visto anteriormente existen dos tipos de ficheros: post-procesado y post-comparación.

	post-procesado		*post-comparación*		***Tiempo total peor caso***
	CG	**DTG**	**Mejor caso**	**Peor Caso**	
Logistics	0,01 (0,01)	0,52 (0,42)	0,07	24,8	25,44
Parking	0,06 (0,04)	1,15 (0,4)	18,3	1980	1981,65
Storage	0,01 (0,01)	0,1 (0,2)	0,02	21,4	22,62

Cuadro 4.1: Tiempos de generación de ficheros

El tiempo de generación de los grafos es reducido, que tiene cómo mínimo un total de $0,11$ segundos en el caso de *storage* y un valor máximo de 2,02 segundos en el peor caso del *parking*. Sin embargo, el caso de la generación de los ficheros de post-comparación alcanza unos valores altos. Esto es debido a la complejidad computacional que tienen los algoritmos de comparación, especialmente el que se ha diseñado para el DTG. Esto hace que se alcance un tiempo en el peor caso de 33 minutos (1980 segundos) en la comparación de los dos problemas más grandes de ese dominio *parking*.

4.2. Explotación del sistema

Para la ejecución sistemática del sistema se han usado un total de 90 problemas para evaluar sus resultados divido de la siguiente manera:

- 30 problemas *logistics*: Formados en grupos de 5 problemas con la configuración que aparece en la tabla 4.2.

Problemas	Aviones	Ciudades	Tamaño ciudad	paquetes
1 - 5	2	3	2	10
6 - 10	2	3	4	10
11 -15	2	3	4	20
16 - 20	3	6	2	30
21 - 25	3	6	4	30
25 - 30	3	6	4	50

Cuadro 4.2: Configuración problemas *logistics*

- 30 problemas *parking*: Formados en grupos de 5 problemas con la configuración que aparece en la tabla 4.3.

Problemas	Bordillos	Coches
1 - 5	5	8
6 - 10	6	10
11 -15	7	12
16 - 20	8	14
21 - 25	9	16
25 - 30	10	18

Cuadro 4.3: Configuración problemas *parking*

- 30 problemas *storage*: Formados en grupos de 5 problemas con la configuración que aparece en la tabla 4.4.

Problemas	Contenedores	Cajas	Grúas	Áreas	Almacenes
1 - 5	1	2	1	4	1
6 - 10	1	3	1	6	1
11 -15	1	4	1	8	1
16 - 20	1	4	2	8	1
21 - 25	1	4	2	12	2
25 - 30	1	6	2	12	2

Cuadro 4.4: Configuración problemas *storage*

4.3. Técnicas de análisis

Después de haber obtenido todos los datos en la fase de explotación, se ha realizado diversas técnicas de análisis de datos para obtener información, Para ello se ha utilizado el programa Weka 3.6, pasando los datos al formato adecuado como se puede ver en el apéndice 2 apartado B.2.

Se han realiza un total de cuatro técnicas distintas para poder analizar los datos y obtener conclusiones al respecto. Las técnicas son:

- **ZeroR**: Se trata de un método supervisado de clasificación que implementa el algoritmo 1R. (A)
- **M5Rules**: Se trata de un método supervisado de predicción que implementa el algoritmo M5. (B)
- **IBK k =1,3,5**: Se trata de un método supervisado de aprendizaje basado en ejemplares. que utiliza el algoritmo de k-vecinos (KNN). (C)
- **Lineal Regression**: Se trata de un método de predicción lineal, que utiliza la regresión lineal. (D)

4.4. Descripción de los datos

Todos los ficheros de problemas que se van a utilizar tienen 30 entradas de 31 atributos cada uno. Y la prueba general contiene los 30 ficheros con sus 30

entradas; es decir, un total de 900. Esto es debido a que no se considera la diagonal, que se trata de la comparación del problema con él mismo. A continuación se especifican en el orden que aparecen cada uno de los atributos incluidos en el fichero Weka.

1. **numeroProblema**: Nos indica el número de problema que queremos analizar. Además, debido al orden que siguen puede indicar si el tamaño del problema es más pequeño o más grande. Por ejemplo, en el dominio *logistics*, si comparamos dos problemas con valores de 20 a 24 significan que tienen un problema pequeño, correspondientes a la primera fila de la tabla 4.2.

2. **cg**: Valor resultante de la comparación de los dos grafos *causal graph* siguiendo el procedimiento descrito en la sección 3.3.2.

3. **dtg**: Valor resultante de la comparación del conjunto de grafos *domain transition graph* de los dos problemas siguiendo el procedimiento descrito en la sección 3.3.3.

4. **estadosIniciales**: Valor resultante de la comparación de los estados iniciales siguiendo el procedimiento descrito en la sección 3.3.4..

5. **longitudc**: Atributo derivado: $|longitud1 - logitud2|$.

6. **longitud1**: Longitud de la solución del problema 1 en la comparación.

7. **longitud2**: Longitud de la solución del problema 2 en la comparación.

8. **evaluadosc**: Atributo derivado: $|evaluados1 - evaluados2|$.

9. **evaluados1**: Número de estados evaluados del problema 1 en la comparación.

10. **evaluados2**: Número de estados evaluados del problema 2 en la comparación.

11. **generadosc**: Atributo derivado: $|generados1 - generados2|$.

12. **generados1**: Número de estados generados del problema 1 en la comparación.

13. **generados2**: Número de estados generados del problema 2 en la comparación.

14. **heuristicac**: Atributo derivado: $|heuristica1 - heuristica2|$.

15. **heuristica1**: Valor inicial de la heurística en el problema 1.

16. **heuristica2**: Valor inicial de la heurística en el problema 2.

17. **tBusquedac**: Atributo derivado: $tBusqueda1 - tBusqueda2$.

18. **tBusqueda1**: Tiempo que tarda el problema 1 en resolverse.

19. **tBusqueda2**: Tiempo que tarda el problema 2 en resolverse.

20. **tTotalc**: Atributo derivado: $tTotal1 - tTotal2$.

21. **tTotal1**: Tiempo total que ha tardado el planificador usado en mostrar la solución del problema 1.

22. **tTotal2**: Tiempo total que ha tardado el planificador usado en mostrar la solución del problema 2.

23. **metasc**: Atributo derivado: $|metas1 - metas2|$.

24. **metas1**: Atributo que indica el número de literales que existen en la meta en el problema 1.

25. **metas2**: Atributo que indica el número de literales que existen en la meta en el problema 2.

26. **literalesc**: Atributo derivado: $|literales1 - literales2|$.

27. **literales1**: Atributo que indica el número de literales que existen en el estado inicial en el problema 1.

28. **literales2**: Atributo que indica el número de literales que existen en el estado inicial en el problema 2.

29. **objetosc**: Atributo derivado: $|objetos1 - objetos2|$.

30. **objetos1**: Número de objetos que hay en el problema 1.

31. **objetos2**: Número de objetos que hay en el problema 2.

De los atributos anteriores, los que tenemos conocimiento previo a la resolución del problema son: objetos, literales, metas, número de problema, dtg, cg, estados iniciales y la heurística. El resto son obtenidos a partir de la solución del problema.

4.5. Descripción de los ficheros de entrenamiento

Debido a la distinta naturaleza de los atributos se han dividido 7 grupos distintos:

1. **Atributos de entrada del problema 1**, Son todos aquellos que conocemos a priori del problema 1, Es el conjunto de atributos: 30, 27, 24 y 15. A este conjunto le denominaremos $A1$.

2. **Atributos de entrada del problema 2**, Son todos aquellos que conocemos a priori del problema 2. Es el conjunto de atributos: 31, 28, 25 y 16. A este conjunto le denominaremos $A2$.

3. **Atributos de salida del problema 1**, Son todos aquellos que conocemos después de obtener la solución del problema 1. Es el conjunto de atributos: 6, 9, 12, 18, 21. A este conjunto le denominaremos $A3$.

4. **Atributos de salida del problema 2**, Son todos aquellos que conocemos después de obtener la solución del problema 1. Es el conjunto de atributos: 7, 10, 13, 19 y 22. A este conjunto le denominaremos $A4$.

5. **Atributos derivados de entrada**: Resultan de la comparación de los atributos de entrada de los problemas. Es el conjunto: 14, 23, 26 y 29. A este conjunto le denominaremos $A5$.

6. **Atributos derivados de salida**: Resultan de la comparación de los atributos de salida de los problemas. Es el conjunto: 5, 8, 11, 17 y 20. A este conjunto le denominaremos $A6$.

7. **Atributos de comparación de grafos:** Estos atributos resultan de la comparación explicada en el capítulo 4, Es el conjunto: 2, 3 y 4. A este conjunto le denominaremos $A7$.

En la tabla 4.5se muestra un resumen de lo que se ha explicado hasta ahora en este punto.

Conjunto	Atributos
A1: Atributos de entrada del problema 1	30: objetos1
	27: literales1
	24: metas1
	15:heuristica1
A2: Atributos de entrada del problema 2	31: objetos2
	28: literales2
	25: metas2
	16: heuristica2
A3: Atributos de salida del problema 1	6: longitud1
	9: evaluados1
	12: generados1
	18: TBusqueda1
	21: tTotal1
A4: Atributos de salida del problema 2	7: longitud2
	10: evaluados2
	13: generados2
	19: tBusqueda2
	22: tTotal2
A5: Atributos derivados de entrada	14: heuristicac
	23: metasc
	26: literalesc
	29: objetosc
A6: Atributos derivados de salida	5: longitudc
	8: evaluadosc
	11: generadosc
	17: tBusquedac
	20: tTotalc
A7: Atributos comparación de grafos	2: cg
	3: dtg
	4: estadosIniciales

Cuadro 4.5: División de atributos de entrada

El atributo número de problema es descartado por no ofrecer información relevante. la relación de tamaños también se encuentra incluida en la comparación del resto de atributos.

Con esta división de atributos se ha creado un conjunto de datos con la idea de obtener la mayor información posible. La división de las pruebas de los tres dominios es la que aparece en el cuadro 4.6.

Datos de entrada	Datos de salida	Atributo a predecir	Nombre de la prueba
A1	A3	longitud1	longitud1
A1	A3	evaluados1	evaluados1
A1	A4	longitud2	longitud2-sin A7
A1	A4	evaluados2	evaluados2- sin A7
A1	A4	longitud2	longitud2-A7
A1	A4	evaluados2	evaluados2-A7
A1	A6	longitudc	longitudc-sin A7
A1	A6	evaluadosc	evaluadosc-sin A7
A1	A6	longitudc	longitudc-A7
A1	A6	evaluadosc	evaluadosc-A7
A1	A6	longitudc	longitudc-A7,A5
A1	A6	evaluadosc	evaluadosc-A7,A5
A1	A4	longitud2	longitud2-A7,A5
A1	A4	evaluados2	evaluados2-A7,A5

Cuadro 4.6: Conjuntos de datos

Además de esta división, se han creado otros conjunto de datos con la aplicación de dos filtros:

1. **Filtro para quitar los valores fuera de rango:** Este filtro se ha aplicado para que los errores que se muestran no sean demasiado elevados por aquellos datos que se encuentren muy alejados de la media y no sean representativos del conjunto (denominados *out layer*).

2. **Filtro de preselección de datos:** Este filtro selecciona los datos más relevantes de manera automática.

Con lo que por cada conjunto de datos deben existir al menos otros dos ficheros de datos que contengan los filtros anteriores. Para entrar más en detalle, los conjuntos pertinentes vienen más detallados en los apartados en los que se evalúan.

4.6. Valores obtenidos

Los valores que se muestran de las pruebas son:

- **La raíz cuadrada del error relativo**: Es la raíz cuadrada del cociente (la división) entre el error absoluto y el valor exacto. Si se multiplica por 100 se obtiene el tanto por ciento (%) de error. Teniendo en cuenta que el error absoluto es la diferencia entre el valor de la medida y el valor tomado como exacto.

- **La desviación estándar o desviación típica** (σ) es una medida de centralización o dispersión para variables de razón (ratio o cociente) y de intervalo, de gran utilidad en la estadística descriptiva. Se define como la raíz cuadrada de la varianza.

4.7. Selección de variables objetivo

Los ficheros de datos que se han creado poseen un total de cinco valores de salida: el número de nodos evaluados, el número de nodos generados, el tiempo total, el tiempo de búsqueda, y la longitud del plan. Se puede decir que el tiempo total es dependiente del planificador, ya que simplemente se trata de $tBusqueda$ más un incremento de tiempo que dedica el planificador a cálculos internos, con lo que el análisis con esa variable queda descartada desde el primer momento.

Referente al tiempo de búsqueda, los evaluados y generados, puede decirse que son dependientes, ya que aumentando el valor de las últimas dos variables aumenta el tiempo. Con lo que la variable $tBusqueda$ también queda descartada.

De las tres variables que nos quedan dos de ellas tratan de los nodos que se generan en la búsqueda. Sin embargo, los evaluados normalmente suelen estar mas intrínsecamente relacionados con el tiempo que tarda y la complejidad que posee el problema, con lo que una de las variables seleccionadas va a tratarse de la cantidad de nodos que se evalúan ($evaluados1$, $evaluados2$ y $evaluadosc$). Además de esta variable también se ha seleccionado la longitud del plan, porque suele estar relacionada con la complejidad del mismo, ya que un valor elevado del plan puede simbolizar que existen muchos objetos en el dominio, que la situación inicial esta muy alejada del objetivo o que hay que realizar muchas acciones para llegar al final. En cualquiera de los tres casos, se ha considerado relevante para este estudio.

Las pruebas que se han realizado son un total de 18, estando divididas por su función objetivo y dominio. La función puede ser: $evaluados1$, $evaluados2$, $evaluadosc$, $longitud1$, $longitud2$ y $longitudc$ y los dominios son $logistics$, $parking$ y $storage$.

Dentro de cada una de las pruebas existen un conjunto de ficheros donde aparecen los datos completos y aplicando filtros. Los filtros aplicados son los que aparecen en el apartado 4.5, sin embargo la cantidad que se aplica en cada una de las pruebas depende de la estructura propia de los datos. Por norma general, si los datos no están uniformemente distribuidos se aplica el primer filtro tantas veces como sea necesario. Por el contrario si los datos están uniformemente distribuidos no aparecerá ese filtro. El filtro que siempre se aplica a cada uno de los ficheros es el de selección de atributos (filtro 2), que aparecerá en todas las pruebas. En el caso de que existan los dos tipos de filtros, aparecerá un tercer fichero de datos con la combinación de los dos. De esta manera, el número de ficheros mínimo siempre es 2.

4.8. *Logistics*

En este apartado se muestran los resultados de las pruebas realizadas con los problemas del dominio *logistics*.

4.8.1. Función objetivo: evaluados1

Los conjuntos de datos que se utilizan para *evaluados*1 se encuentran divididos en:

1. El conjunto de datos completo. Atributos de entrada $A1$. (1)
2. El conjunto de datos *evaluados*1 eliminando los valores por encima de 4500 (870/900 instancias). Atributos de entrada A1. (2)
3. El conjunto *evaluados*1 aplicando selección de atributos. Atributo de entrada: 15. (3)
4. El conjunto *evaluados*1 aplicando selección de atributos y restringiendo los valores por encima de 4500 (870/900 instancias). Atributo de entrada: 15. (4)

	ZeroR	M5Rules	IBK k =1	IBK k=3	IBK k=5	LinearRegression
(1)	100(0,00)	77.83(16.85)	84.70(11.18)	84.70(11.18)	84.70(11.18)	77,71(16,76)
(2)	100(0,00)	42.21(21.21)	**41.92(20.64)**	**41.92(20.64)**	**41.92(20.64)**	52,50(14,74)
(3)	100(0,00)	78,91(15,56)	85,41(10,54)	85,41(10,54)	85,41(10,54)	78,91(15,56)
(4)	100(0,00)	46,24(18,75)	45,73(18,20)	45,73(18,20)	45,73(18,20)	45,73(18,20)

Cuadro 4.7: Raíz cuadrada del error relativo medio y desviación típica de *Logistics* Evaluados1

Como se puede ver en la tabla en conjunto de datos (2) obtiene el mejor resultado en todos los casos, lo que quiere decir que para predecir el número de evaluados se necesitan todos los atributos, pero se quitan los datos fuera de rango. Si observamos el valor de (4) podemos ver que el valor es muy similar, sólo aumenta en 4 su valor y su desviación típica desciende y con este caso sólo es necesario conocer el valor que se obtiene en la heurística eliminando los valores fuera de rango. Este resultado es muy acertado porque el valor de la heurística estima la longitud para llegar al objetivo, y a menor valor de la heurística menor valor de evaluados.

4.8.2. Función objetivo: evaluados2

Los conjuntos de datos que se utilizan para $evaluados2 - A7$, $evaluados2 - sinA7$ y $evaluados2 - A7, A5$ se encuentran divididos en:

1. El conjunto de datos $evaluados2 - sinA7$. (1)
2. El conjunto de datos $evaluados2 - sinA7$ eliminando los valores por encima de 2800 (870/900 instancias). (2)
3. El conjunto de datos $evaluados2 - sinA7$ aplicando selección de atributos. Atributos de entrada: 15 y 31. (3)
4. El conjunto de datos $evaluados2 - sinA7$ aplicando selección de atributos y eliminando los valores por encima de 2800 (870/900 instancias). Atributos de entrada: 12 y 16. (4)

5. El conjunto de datos $evaluados2 - A7$. (5)

6. El conjunto de datos $evaluados2 - A7$ eliminando los valores por encima de 2800(870/900 instancias) (6)

7. El conjunto de datos $evaluados2 - A7$ aplicando selección de atributos. Atributos de entrada: 16 , 24, 28 y 31. (7)

8. El conjunto de datos $evaluados2 - A7$ aplicando selección de atributos y eliminando los valores por encima de 2800 (870/900 instancias). Atributos de entrada: 12 y 16. (8)

9. El conjunto de datos $evaluados2 - A7, A5$. (9)

10. El conjunto de datos $evaluados2 - A7, A5$ eliminando los valores por encima de 2800 (870/900 instancias). (10)

11. El conjunto de datos $evaluados2 - A7, A5$ aplicando selección de atributos. Atributos de entrada: 16 ,24, 28 y 31. (11)

12. El conjunto de datos $evaluados2 - A7, A5$ aplicando selección de atributos y eliminando los valores por encima de 2800 (870/900 instancias). Atributos de entrada: 15 y 16. (12)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	16,22(6,90)	154,01(42,39)	101,77(24,22)	96,13(17,79)	87,48(3,35)
(2)	100(0,0)	16,32(3,06)	98,52(13,71)	68,26(8,64)	66,16(7,66)	64,15(8,67)
(3)	100(0,0)	78,99(15,59)	103,13(19,93)	103,05(20,14)	97,42(17,15)	78,99(15,59)
(4)	100(0,0)	22,12(2,92)	34,99(7,54)	25,32(5,74)	25,27(5,20)	68,91(10,25)
(5)	100(0,0)	16,21(6,86)	108,38(28,34)	84,64(17,56)	84,05(14,84)	87,59(4,13)
(6)	100(0,0)	16,35(3,01)	93,18(14,03)	69,65(9,76)	67,46(8,27)	63,77(8,72)
(7)	100(0,0)	16,63(6,56)	**3,60(1,28)**	6,82(10,87)	55,28(16,12)	90,63(2,84)
(8)	100(0,0)	22,12(2,92)	34,99(7,54)	25,32(5,74)	25,27(5,20)	68,91(10,25)
(9)	100(0,0)	16,30(6,82)	117,81(29,48)	90,75(17,69)	85,13(13,74)	87,96(4,83)
(10)	100(0,0)	16,22(3,00)	94,10(14,37)	72,75(10,15)	68,17(8,27)	63,48(8,69)
(11)	100(0,0)	16,63(6,56)	**3,60(1,28)**	6,82(10,87)	55,28(16,12)	90,63(2,84)
(12)	100(0,0)	22,13(2,92)	57,27(12,59)	57,02(9,26)	60,34(7,76)	69,16(10,34)

Cuadro 4.8: Raíz cuadrada del error relativo medio y desviación típica de *Logistics* evaluados2

Los mejores resultados obtenidos son los de los ficheros de datos (7) y (11) en ambos casos se aplica la selección de atributos y no deja ningún dato derivado de los grafos. En el caso del grupo de datos (7), los atributos por los que clasifica son $heuristica2$, $metas1$, $literales2$ y $objetos2$.

4.8.3. Función objetivo: evaluadosc

Los conjuntos de datos que se utilizan para $evaluadosc - A7$, $evaluadosc - sinA7$ y $evaluadosc - A7, A5$ se encuentran divididos en:

1. El conjunto de datos $evaluadosc - sinA7$. (1)

2. El conjunto de datos $evaluadosc - sinA7$ eliminando los valores por encima de 2800 (859/900 instancias). (2)

3. El conjunto de datos $evaluadosc - sinA7$ aplicando selección de atributos. Atributos de entrada: 9, 16, 25, 28 y 31. (3)

4. El conjunto de datos $evaluadosc - sinA7$ aplicando selección de atributos 7 eliminando los valores por encima de 2800 (859/900 instancias). Atributos de entrada: 9, 16, 25, 28 y 31. (4)

5. El conjunto de datos $evaluadosc - A7$. (5)

6. El conjunto de datos $evaluadosc - A7$ eliminando los valores por encima de 4100 (862/900 instancias). (6)

7. El conjunto de datos $evaluadosc - A7$ aplicando selección de atributos. Atributos de entrada: 16 ,24, 28 y 31. (7)

8. El conjunto de datos $evaluadosc - A7$ aplicando selección de atributos eliminando los valores por encima de 4100 (862/900 instancias) Atributos de entrada: 16 ,24, 28 y 31. (8)

9. El conjunto de datos $evaluadosc - A7$ eliminando los valores por encima de 8300 (863/870 instancias) (9)

10. El conjunto de datos $evaluadosc - A7$ aplicando selección de atributos eliminando los valores por encima de 8300 (893/900 instancias) Atributos de entrada: 2, 3, 9, 16, 21, 28 y 31. (10)

11. El conjunto de datos $evaluadosc - A7, A5$. (11)

12. El conjunto de datos $evaluadosc - A7, A5$ eliminando los valores por encima de 4100 (862/900 instancias). (12)

13. El conjunto de datos $evaluadosc - A7, A5$ aplicando selección de atributos. Atributos de entrada: 14, 16 y 24. (13)

14. El conjunto de datos $evaluadosc - A7, A5$ eliminando los valores por encima de 8300 (893/900 instancias). (14)

15. El conjunto de datos $evaluadosc - A7, A5$ aplicando selección de atributos. Atributos de entrada: 4, 14, 17, 28 y 31. (15)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	70,35(284,50)	143,60(40,39)	102,66(20,21)	98,83(13,48)	77,33(21,53)
(2)	100(0,0)	48,12(26,11)	100,70(16,42)	68,14(10,19)	67,55(8,24)	91,82(4,16)
(3)	100(0,0)	**27,65(13,53)**	35,16(25,13)	38,38(24,99)	41,66(25,18)	80,03(21,19)
(4)	100(0,0)	52,71(116,23)	32,67(10,48)	35,34(7,83)	41,40(6,80)	92,45(4,09)
(5)	100(0,0)	36,20(31,40)	108,00(29,13)	88,97(14,14)	89,54(11,62)	75,71(21,23)
(6)	100(0,0)	44,85(83,05)	106,62(36,02)	84,84(21,68)	86,15(18,91)	93,04(5,76)
(7)	100(0,0)	39,95(15,60)	64,85(27,60)	60,19(19,82)	72,67(22,65)	96,43(3,94)
(8)	100(0,0)	82,49(282,09)	90,31(12,57)	68,12(10,32)	71,18(8,70)	79,73(7,63)
(9)	100(0,0)	94,08(388,49)	57,98(11,57)	54,24(8,55)	59,47(7,82)	84,21(6,56)
(10)	100(0,0)	47,11(45,67)	115,08(30,13)	93,54(14,37)	90,35(10,49)	75,43(21,30)
(11)	100(0,0)	55,02(36,43)	92,04(13,94)	73,23(10,85)	70,91(9,08)	67,80(9,64)
(12)	100(0,0)	35,04(13,88)	100,99(40,96)	85,85(27,19)	81,24(19,80)	95,50(3,23)
(13)	100(0,0)	59,13(8,81)	82,34(10,54)	76,44(8,51)	75,58(7,63)	75,49(9,40)
(14)	100(0,0)	53,08(119,97)	112,10(35,88)	91,24(20,82)	88,49(18,04)	92,57(5,12)
(15)	100(0,0)	31,56(16,33)	101,21(26,92)	86,46(14,98)	86,86(10,93)	78,70(20,87)

Cuadro 4.9: Raíz cuadrada del error relativo medio y desviación típica de *logistics* evaluadosc

Los mejores resultados se obtienen con el fichero de datos (3) y la técnica M5Rules, este fichero no tiene en cuenta los valores de los grafos. Además tiene aplicada una selección de atributos dejando solamente $evaluados1$, $heuristica2$, $metas2$, $literales2$ y $metas2$, prácticamente todos los atributos de entrada del problema 2.

4.8.4. Función objetivo: longitud1

Los conjuntos de datos que se utilizan para $longitud1$ se encuentran divididos en:

1. El conjunto de datos completo $longitud1$. Atributos de entrada $A1$. (1)
2. El conjunto de datos $longitud1$ eliminando los valores por encima de 180 (888/900 instancias). Atributos de entrada A1. (2)
3. El conjunto $longitud1$ aplicando selección de atributos. Atributo de entrada: 15, 27 y 30. (3)
4. El conjunto $longitud1$ aplicando selección de atributos y restringiendo los valores por encima de 180 (888/900 instancias). Atributo de entrada: 15. (4)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	85,84(13,91)	89,98(11,18)	89,98(11,18)	89,98(11,18)	84,12(15,54)
(2)	100(0,0)	3,81(0,47)	**2,42(0,51)**	**2,42(0,51)**	**2,42(0,51)**	9,25(0,92)
(3)	100(0,0)	85,82(13,89)	89,98(11,18)	89,98(11,18)	89,98(11,18)	83,20(18,40)
(4)	100(0,0)	4,79(0,54)	3,86(0,63)	3,86(0,63)	3,86(0,63)	10,45(1,32)

Cuadro 4.10: Raíz cuadrada del error relativo medio y desviación típica de *Logistics* longitud1

Los mejores resultados se obtienen en el fichero (2) en IBK, esto ocurre porque existían unos pocos valores fuera del rango de la media que perjudicaban al clasificador, y se han usado todos los atributos de entrada para predecir la longitud.

4.8.5. Función objetivo: longitud2

Los conjuntos de datos que se utilizan para $longitud2-sinA7$, $longitud2-A7$ y $longitud2-A7, A5$ se encuentran divididos en:

1. El conjunto de datos completo. Atributos de $longitud2-sinA7$. (1)
2. El conjunto $longitud2-sinA7$ aplicando selección de atributos. Atributo de entrada: 9 y 16. (2)
3. El conjunto de datos completo. Atributos de $longitud2-A7$. (3)
4. El conjunto $longitud2-A7$ aplicando selección de atributos. Atributo de entrada: 2, 25 y 28. (4)
5. El conjunto de datos completo. Atributos de $longitud2-A7, A5$. (5)
6. El conjunto $longitud2-A7, A5$ aplicando selección de atributos. Atributo de entrada: 3, 16 y 18. (6)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	70,34(84,10)	50,24(5,09)	48,50(4,56)	46,03(4,18)	99,07(1,47)
(2)	100(0,0)	44,84(7,15)	49,41(4,99)	47,80(4,55)	45,32(4,24)	99,27(1,38)
(3)	100(0,0)	171,79(1004,23)	28,45(3,56)	**26,57(2,87)**	29,97(2,81)	67,61(4,97)
(4)	100(0,0)	726,13(2126,92)	67,67(6,85)	61,01(5,36)	58,03(4,91)	70,06(4,43)
(5)	100(0,0)	60,83(35,78)	54,58(4,67)	45,20(4,39)	44,07(4,11)	51,41(3,89)
(6)	100(0,0)	50,40(4,63)	46,90(4,45)	46,56(4,44)	45,07(3,98)	53,38(4,23)

Cuadro 4.11: Raíz cuadrada del error relativo medio y desviación típica de *Logistics* longitud2

Los mejores resultados los encontramos en el fichero (3) que son con todos los atributos de entrada y los atributos derivados de los grafos, sin incluir los derivados y sin aplicar ningún tipo de filtro. En este caso debido a la inclusión de las comparaciones entre los grafos de los dos problemas ha mejorado casi un 50 % del mejor resultado sin incluirlos. Como se puede ver los resultados del fichero (4) cuando se aplica un filtro y este quita los atributos de los grafos aumenta el error.

4.8.6. Función objetivo: longitudc

Los conjuntos de datos que se utilizan para $longitudc-sinA7$, $longitudc-A7$ y $longitudc-A7, A5$ se encuentran divididos en:

1. El conjunto de datos completo. Atributos de $longitudc-sinA7$. (1)

2. El conjunto $longitudc - sinA7$ aplicando selección de atributos. Atributo de entrada: 9, 12, 15, 18, 21, 25, 28, 30 y 31. (2)

3. El conjunto de datos completo. Atributos de $longitudc - A7$. (3)

4. El conjunto $longitudc - A7$ aplicando selección de atributos. Atributo de entrada: 3, 18 y 21. (4)

5. El conjunto de datos completo. Atributos de $longitudc - A7, A5$. (5)

6. El conjunto $longitudc - A7, A5$ aplicando selección de atributos. Atributo de entrada: 15, 24, 26 y 27. (6)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	33,90(3,13)	48,87(4,27)	40,15(3,74)	37,55(3,42)	35,34(3,48)
(2)	100(0,0)	33,29(2,90)	34,48(3,05)	34,48(3,05)	34,48(3,05)	34,53(3,40)
(3)	100(0,0)	70,34(84,10)	50,24(5,09)	48,50(4,56)	46,03(4,18)	99,07(1,47)
(4)	100(0,0)	44,84(7,15)	49,41(4,99)	47,80(4,55)	45,32(4,24)	99,27(1,38)
(5)	100(0,0)	**3,77(0,38)**	17,99(2,01)	16,84(1,60)	18,82(1,75)	9,31(0,80)
(6)	100(0,0)	7,91(11,55)	8,86(2,49)	8,27(1,72)	8,65(1,36)	10,50(1,15)

Cuadro 4.12: Raíz cuadrada del error relativo medio y desviación típica de *Logistics* longitudc

Los mejores resultados los encontramos en el fichero (5) y fichero (6) que son con todos los atributos de entrada incluyendo los derivados y en el último caso aplicando un filtro de selección de atributos. Si se tiene en cuenta en el fichero (5), que aparecen todos los atributos relacionados con los grafos, se obtienen mejores datos en 2 de las 6 pruebas (M5Rules y linealRegression), sin embargo, en el fichero (6), que tiene aplicado un filtro de selección de atributos y no se encuentran todos, tiene mejores resultados para IBK. Aunque la diferencia de valores de $3,77$ del fichero (5) al mejor del valor (6) es considerable ya que posee un valor de $8,65$, con lo que la comparación realizada con los grafos aporta la suficiente información como para determinar una mejor solución.

4.8.7. Análisis deresultados

En este dominio no existe una relación clara entre los buenos resultados en la predicción de las variables de salida y la existencia de las variable de entrada de los grafos. Esto se debe a la gran similitud que existen entre los grafos de este dominio. Lo que conlleva a que otras variables poseen una carga superior.

En casi todos los casos, excluyendo los dos últimos, no han aparecido variables derivadas de los grafos, siendo los atributos más importantes para realizar la predicción la heurística del primer problema ($heuristica1$), la heurística del segundo problema ($heuristica2$), objetos del segundo problema ($objetos2$), metas del segundo problema ($metas2$) y los literales del segundo problema ($literales2$). La selección de estos atributos puede resultar la indicada porque de manera habitual, cuando en un problema se aumentan cada uno de los últimos elementos suele aumentar la complejidad y el tamaño de la solución.

Para ver más claro en que proporción influyen se muestra una recta de regresión para el mejor resultado obtenido en *evaluados*1.

$$evaluados1 = 21{,}9313 * heuristica1 - 19{,}7704 * objetos1 - 183{,}5774$$

Como se ha comentado en este caso, debido a que no existen atributos del problema dos, la carga de la solución depende del valor de la heurística y de los *objetos*1.

Para la predicción de *evaluados*2 que se ha encontrado la mejor solución en M5Rules. El modelo resultante tiene un total de 9 reglas en las que solo se tiene en cuenta la *heuristica*2 y los *literales*2.

Para el caso de *longitudc* también se ha encontrado la mejor solución en M5Rules. Se han conseguido un total de 11 reglas donde los atributos con más peso son *literalesc*, *literales*1, *cg* y *estadosiniciales*.

4.9. *Parking*

En este apartado se muestran los resultado de las pruebas realizadas con los problemas del dominio *parking*.

4.9.1. Función objetivo: evaluados1

Los conjuntos de datos que se utilizan para *evaluados*1 se encuentran divididos en:

1. El conjunto de datos completo *evaluados*1. Atributos de entrada *A*1. (1)
2. El conjunto de datos *evaluados*1 eliminando los valores por encima de 180 (888/900 instancias).(2)
3. El conjunto *evaluados*1 aplicando selección de atributos. Atributo de entrada: 15. (3)
4. El conjunto *evaluados*1 aplicando selección de atributos y restringiendo los valores por encima de 180 (888/900 instancias). Atributo de entrada: 15. (4)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	47,81(12,36)	**46,85(12,50)**	**46,85(12,50)**	**46,85(12,50)**	91,08(5,57)
(2)	100(0,0)	61,15(13,57)	60,57(14,45)	60,57(14,45)	60,57(14,45)	91,52(4,99)
(3)	100(0,0)	86,59(11,31)	87,86(11,12)	87,86(11,12)	87,86(11,12)	98,23(11,15)
(4)	100(0,0)	87,45(23,20)	81,59(26,78)	81,59(26,78)	81,59(26,78)	97,18(9,06)

Cuadro 4.13: Raíz cuadrada del error relativo medio y desviación típica de *parking* evaluados1

Los mejores resultados se han obtenido en el fichero (1), en este fichero aparecen todos los atributos de entrada (*A*1) y se ha hecho una limpieza de los datos fuera de rango. Sin embargo, aunque los datos mejoran, se puede observar por los resultados que no son lo suficientemente buenos como para dar una aproximación del valor real.

4.9.2. Función objetivo: evaluados2

Los conjuntos de datos que se utilizan para $evaluados2 - A7$, $evaluados2 - sinA7$ y $evaluados2 - A7, A5$ se encuentran divididos en:

1. El conjunto de datos $evaluados2 - sinA7$. (1)

2. El conjunto de datos $evaluados2 - sinA7$ eliminando los valores por encima de 29200 (897/900 instancias). (2)

3. El conjunto de datos $evaluados2 - sinA7$ aplicando selección de atributos. Atributos de entrada: 9, 16, 18, 25, 28 y 31. (3)

4. El conjunto de datos $evaluados2 - sinA7$ aplicando selección de atributos y eliminando los valores por encima de 29200 (897/900 instancias). Atributos de entrada: 9, 16, 18, 25, 28 y 31. (4)

5. El conjunto de datos $evaluados2 - A7$. (5)

6. El conjunto de datos $evaluados2 - A7$ eliminando los valores por encima de 29200 (897/900 instancias) (6)

7. El conjunto de datos $evaluados2 - A7$ aplicando selección de atributos. Atributos de entrada: 4, 9, 16, 18, 25 , 28 y 31. (7)

8. El conjunto de datos $evaluados2 - A7$ aplicando selección de atributos y eliminando los valores por encima de 29200 (897/900 instancias). Atributos de entrada:4, 9, 16, 18, 25 , 28 y 31. (8)

9. El conjunto de datos $evaluados2 - A7, A5$. (9)

10. El conjunto de datos $evaluados2 - A7, A5$ eliminando los valores por encima de 29200 (897/900 instancias) (10)

11. El conjunto de datos $evaluados2 - A7, A5$ aplicando selección de atributos. Atributos de entrada: 4, 9, 14, 16, 18, 25, 26, 28, 29 y 31. (11)

12. El conjunto de datos $evaluados2 - A7, A5$ aplicando selección de atributos y eliminando los valores por encima de 29200 (897/900 instancias). Atributos de entrada: 4, 9, 14, 16, 18, 23, 25, 26, 28, 29 y 31. (12)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	42,75(9,92)	90,58(16,24)	75,76(14,97)	70,73(12,61)	89,89(5,50)
(2)	100(0,0)	32,12(4,35)	112,34(13,01)	83,62(8,99)	82,45(6,42)	96,44(4,15)
(3)	100(0,0)	55,29(13,28)	83,67(19,68)	67,53(15,20)	64,32(15,62)	90,65(5,10)
(4)	100(0,0)	86,12(5,00)	115,72(7,78)	106,36(6,65)	96,67(6,19)	98,16(2,38)
(5)	100(0,0)	43,42(10,50)	88,75(19,46)	80,14(14,75)	73,96(12,40)	90,46(6,16)
(6)	100(0,0)	**31,86(5,14)**	106,62(12,89)	87,55(8,66)	87,26(6,88)	95,81(4,16)
(7)	100(0,0)	55,29(13,28	83,67(19,68)	67,53(15,20)	64,32(15,62)	90,65(5,10)
(8)	100(0,0)	86,12(5,00)	115,72(7,78)	106,36(6,65)	96,67(6,19)	98,16(2,38)
(9)	100(0,0)	43,21(10,29)	87,40(18,86)	72,51(15,41)	69,29(15,04)	87,89(7,15)
(10)	100(0,0)	33,53(6,71)	104,18(12,25)	84,37(9,14)	82,58(6,91)	93,31(6,29)
(11)	100(0,0)	56,08(13,70)	79,47(18,95)	64,32(15,82)	63,00(16,26)	88,91(6,11)
(12)	100(0,0)	87,16(5,69)	90,82(9,83)	88,71(7,58)	89,28(6,94)	96,88(3,43)

Cuadro 4.14: Raíz cuadrada del error relativo medio y desviación típica de *parking* evaluados2

En esta prueba los datos son más dispares debido a que *evaluados*2 no se trata de un atributo de salida directo. Sin embargo la solución obtenida en el fichero (6) obtiene mejor resultado que la prueba anterior, consiguiendo un valor de $31,86$. Este fichero toma como datos de entrada los conjuntos ($A1$, $A2$, $A3$ y $A7$) en el que está incluidos los relacionados con los grafos.

4.9.3. Función objetivo: evaluadosc

Los conjuntos de datos que se utilizan para $evaluadosc - A7$, $evaluadosc - sinA7$ y $evaluadosc - A7, A5$ se encuentran divididos en:

1. El conjunto de datos $evaluadosc - sinA7$. (1)
2. El conjunto de datos $evaluadosc - sinA7$ eliminando los valores por encima de 29200 (897/900 instancias). (2)
3. El conjunto de datos $evaluadosc - sinA7$ aplicando selección de atributos. Atributos de entrada: 12 y 16. (3)
4. El conjunto de datos $evaluadosc - sinA7$ aplicando selección de atributos eliminando los valores por encima de 29200 (897/900 instancias)., Atributos de entrada: 9 y 31. (4)
5. El conjunto de datos $evaluadosc - A7$. (1)
6. El conjunto de datos $evaluadosc - A7$ eliminando los valores por encima de 2200 (897/900 instancias). (2)
7. El conjunto de datos $evaluadosc - A7$ aplicando selección de atributos. Atributos de entrada: 12 y 16. (3)
8. El conjunto de datos $evaluadosc - A7$ aplicando selección de atributos eliminando los valores por encima de 29200 (897/900 instancias)., Atributos de entrada: 9 y 31. (4)
9. El conjunto de datos $evaluadosc - A7, A5$. (1)

10. El conjunto de datos $evaluadosc - A7, A5$ eliminando los valores por encima de 29200 (897/900 instancias). (2)

11. El conjunto de datos $evaluadosc - A7, A5$ aplicando selección de atributos. Atributos de entrada: 14, 16, 21 y 31. (3)

12. El conjunto de datos $evaluadosc - A7, A5$ aplicando selección de atributos eliminando los valores por encima de 29200 (897/900 instancias). Atributos de entrada: 14 y 31. (4)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	35,28 (17,45)	58,76(24,51)	60,08(11,94)	59,86(10,60)	57,03(25,13)
(2)	100(0,0)	38,70(13,38)	71,00(12,56)	59,63(10,23)	55,62(8,87)	71,12(8,08)
(3)	100(0,0)	**34,78(17,77)**	49,55(21,25)	48,47(11,48)	53,17(13,36)	56,78(25,02)
(4)	100(0,0)	39,23(14,69)	58,58(13,65)	44,17(9,70)	41,09(8,21)	71,31(7,99)
(5)	100(0,0)	**36,26(15,88)**	57,56(25,13)	60,73(12,32)	60,55(10,23)	53,78(23,29)
(6)	100(0,0)	38,62(10,44)	70,09(12,90)	63,04(11,72)	57,92(9,71)	68,13(7,12)
(7)	100(0,0)	39,46(18,15)	55,72(24,77)	59,47(12,35)	64,57(10,46)	55,94(24,36)
(8)	100(0,0)	43,22(15,37)	54,23(13,27)	44,72(10,41)	42,35(9,49)	69,76(7,52)
(9)	100(0,0)	36,80(15,44)	62,77(24,07)	61,59(11,58)	59,54(10,96)	51,66(22,27)
(10)	100(0,0)	36,80(15,44)	62,77(24,07)	61,59(11,58)	59,54(10,96)	51,66(22,27)
(11)	100(0,0)	47,83(46,19)	70,03(25,64)	67,95(10,58)	69,13(8,62)	54,77(23,75)
(12)	100(0,0)	43,22(17,56)	61,46(14,45)	55,73(12,68)	55,75(10,51)	67,63(7,15)

Cuadro 4.15: Raíz cuadrada del error relativo medio y desviación típica de *parking* evaluadosc

En esta prueba los datos han mejorado notablemente en comparación con la prueba anterior (longitud2), obteniendo el mejor resultado en el fichero (3) con $34,78$, y el segundo en el fichero (5) con $36,26$, este fichero de datos corresponde con el que tiene los datos de entrada de los dos problemas ($A1$ y $A2$) y los relativos a los grafos ($A7$). La diferencia entre ambos ficheros es muy pequeña, ya que el fichero (3) tiene más desviación típica que el fichero (6).

Como esta diferencia no es perceptible se ha realizado un test de significación estadística con los resultados obtenidos de *M5Rules*. Y se ha obtenido que todos los valores obtenidos no obtienen diferencia significativa en ninguno de los ficheros. Para el caso de *IBK* el mejor fichero es el (4) y para *linealRegression* el mejor fichero es el (5).

4.9.4. Función objetivo: longitud1

Los conjuntos de datos que se utilizan para $longitud1$ se encuentran divididos en:

1. El conjunto de datos $longitud1$ completo. Atributos de entrada $A1$. (1)

2. El conjunto $longitud1$ aplicando selección de atributos. Atributo de entrada: 15. (3)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	21,47(2,63)	**17,02(2,49)**	**17,02(2,49)**	**17,02(2,49)**	62,89(3,89)
(2)	100(0,0)	32,90(2,49)	30,72(2,23)	30,72(2,23)	30,72(2,23)	64,89(4,31)

Cuadro 4.16: Raíz cuadrada del error relativo medio y desviación típica de *parking* longitud1

Los mejores resultados se obtienen con el fichero de datos (1), en este fichero se tienen todos los datos de entrada ($A1$). Es decir, la aplicación automática del filtro de datos no resulta útil y con el simple dato de la *heuristica*1 se puede predecir correctamente.

4.9.5. Función objetivo: longitud2

Los conjuntos de datos que se utilizan para $longitud2 - sinA7$, $longitud2 - A7$ y $longitud2 - A7, A5$ se encuentran divididos en:

1. El conjunto de datos completo. Atributos de $longitud2 - sinA7$. (1)
2. El conjunto $longitud2 - sinA7$ aplicando selección de atributos. Atributo de entrada: 9 y 16. (2)
3. El conjunto de datos completo. Atributos de $longitud2 - A7$. (3)
4. El conjunto $longitud2 - A7$ aplicando selección de atributos. Atributo de entrada: 9 y 31. (4)
5. El conjunto de datos completo. Atributos de $longitud2 - A7, A5$. (5)
6. El conjunto $longitud2 - A7, A5$ aplicando selección de atributos. Atributo de entrada: 14 y 16. (6)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	21,16(3,83)	84,18(9,72)	66,12(7,65)	60,46(6,07)	64,52(4,19)
(2)	100(0,0)	32,05(2,59)	51,49(4,88)	39,22(3,03)	37,85(2,85)	65,83(4,56)
(3)	100(0,0)	53,52(7,90)	90,30(8,71)	73,69(6,56)	69,80(5,87)	72,37(5,13)
(4)	100(0,0)	62,52(5,51)	79,11(7,19)	77,98(7,03)	69,34(6,04)	72,06(4,57)
(5)	100(0,0)	**20,80(3,21)**	79,27(8,52)	65,83(6,68)	62,37(5,71)	63,92(4,81)
(6)	100(0,0)	32,06(2,59)	48,00(6,86)	53,73(7,95)	55,53(6,57)	65,29(4,90)

Cuadro 4.17: Raíz cuadrada del error relativo medio y desviación típica de *parking* longitud2

Los mejores resultados se obtienen con el fichero (5), en este fichero se encuentran como datos de entrada ($A1$, $A2$, $A3$, $A5$ y $A7$), como los datos están uniformemente distribuidos, no se ha necesitado la aplicación de ningún filtro de datos para eliminar los datos fuera de rango. Con el resultado de $20,80$ lo que nos lleva a concluir es que los atributos relacionados con los grafos $A7$ y los derivados de entrada ($A5$) son importantes para el atributo *longitud*2.

4.9.6. Función objetivo: longitudc

Los conjuntos de datos que se utilizan para $longitudc-sinA7$, $longitudc-A7$ y $longitudc-A7, A5$ se encuentran divididos en:

1. El conjunto de datos completo. Atributos de $longitudc-sinA7$. (1)
2. El conjunto $longitudc-sinA7$ aplicando selección de atributos. Atributo de entrada: 6, 9, 12, 16, 18, 25, 28 y 31. (2)
3. El conjunto de datos completo. Atributos de $longitudc-A7$. (3)
4. El conjunto $longitudc-A7$ aplicando selección de atributos. Atributo de entrada: 2, 3, 9 y 12. (4)
5. El conjunto de datos completo. Atributos de$longitudc-A7, A5$. (5)
6. El conjunto $longitudc-A7, A5$ aplicando selección de atributos. Atributo de entrada: 4, 6, 14, 16 y 26. (6)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	140,45(629,23)	106,28(15,56)	81,96(7,26)	99,36(8,02)	95,50(3,04)
(2)	100(0,0)	58,02(62,11)	80,93(18,98)	70,05(11,97)	71,00(10,50)	96,25(3,05)
(3)	100(0,0)	94,13(207,19)	113,25(13,83)	92,18(7,91)	88,83(5,90)	94,43(3,87)
(4)	100(0,0)	78,12(11,55)	99,37(11,52)	89,98(10,06)	87,67(8,51)	96,08(2,83)
(5)	100(0,0)	**59,22(20,46)**	113,66(16,03)	91,77(7,61)	89,57(6,55)	92,74(4,64)
(6)	100(0,0)	62,01(11,60)	119,94(16,70)	99,36(8,02)	94,93(6,20)	93,77(4,38)

Cuadro 4.18: Raíz cuadrada del error relativo medio y desviación típica de *parking* longitudc

Los mejores resultados se encuentran en el fichero (5), aunque no poseen el menor valor mínimo, posee una desviación mínima mucho menor. Estamos hablando de un $58,02$ $(62,11)$ frente a un $59,22$ $(20,46)$. Con este segundo resultado en media obtenemos mejores valores que con el primero.

El conjunto de datos que ha obtenido mejores resultados, es el que tiene todos los datos de entrada, incluidos los derivados y los relacionados con los grafos ($A1$, $A2$, $A3$, $A5$ y $A7$). Esto quiere decir que estos últimos grupos de atributos son relevantes para el atributo $longituc$, como ha ocurrido con el $longitud2$.

4.9.7. Análisis de resultados

En este dominio, en comparación con el primero, es vital la aparición de los grafos, porque en todos los casos que han aparecido han ayudado a resolver mejor el problema que ocupaba. Por otra parte, es más lógico porque los grafos de estos dominios son menos similares unos a otros que en el caso anterior. Esto conlleva a que si aporten una información relevante para realizar un modelo.

Los atributos más relevantes en este dominio son: la heurística del primer problema ($heuristica1$), la heurística del segundo problema ($heuristica2$), las metas del segundo problema ($metas2$), y los literales del segundo problema ($literales2$). Aunque estos son los que mas relevancia tienen, también se usan la

comparación de los grafos *causal graph* (cg), los estados iniciales ($estadosIniciales$) y la comparación de los grafos *domain transition graph* (dtg). Estos resultados se pueden mostrar en el conjunto de reglas de los mejores resultados. Donde muchas reglas de los nodos hoja tienen fórmulas parecidas a la siguiente:

$$longitud2 = -0{,}6145 * heuristicac + 29{,}2502$$

en el caso de nodos en la parte inferior del árbol y

$$longitud2 = 0 * cg + 0{,}0085 * estadosIniciales + 0{,}1804 * heuristica2 - 0{,}1526 * literales2 + 28{,}9559$$

en nodos superiores del árbol.

4.10. *Storage*

En este apartado se muestran los resultado de las pruebas realizadas con los problemas del dominio *storage*.

4.10.1. Función objetivo: evaluados1

Los conjuntos de datos que se utilizan para $evaluados1$ se encuentran divididos en:

1. El conjunto de datos completo $evaluados1$. Atributos de entrada $A1$. (1)
2. El conjunto de datos $evaluados1$ eliminando los valores por encima de 180 (893/900 instancias). Atributos de entrada A1. (2)
3. El conjunto $evaluados1$ aplicando selección de atributos. Atributo de entrada: 30. (3)
4. El conjunto $evaluados1$ aplicando selección de atributos y restringiendo los valores por encima de 180 (893/900 instancias). Atributo de entrada: 15 y 30. (4)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	60,74(19,77)	62,98(20,76)	60,67(20,07)	60,49(20,13)	86,22(6,56)
(2)	100(0,0)	20,51(6,32)	22,13(7,89)	22,07(8,33)	22,40(8,95)	80,80(3,44)
(3)	100(0,0)	68,49(15,67)	63,67(19,09)	63,67(19,09)	63,67(19,09)	85,30(6,57)
(4)	100(0,0)	20,57(6,24)	**19,18(6,40)**	**19,18(6,40)**	**19,18(6,40)**	79,49(3,59)

Cuadro 4.19: Raíz cuadrada del error relativo medio y desviación típica de *storage* evaluados1

En esta prueba los mejores resultados los ofrecidos por el conjunto (4), lo que significa que solamente con los atributos de la heurística ($heuristica1$) y las objetos ($objetos1$), incluyendo una limpieza de los datos iniciales eliminando las muestras superiores a 180 y se obtiene un error del $19,18$.

4.10.2. Función objetivo: evaluados2

Los conjuntos de datos que se utilizan para $evaluados2 - A7$, $evaluados2 - sinA7$ y $evaluados2 - A7, A5$ se encuentran divididos en:

1. El conjunto de datos $evaluados2 - sinA7$. (1)

2. El conjunto de datos $evaluados2 - sinA7$ eliminando los valores por encima de 2800 (870/900 instancias). (2)

3. El conjunto de datos $evaluados2 - sinA7$ aplicando selección de atributos. Atributos de entrada: 24. (3)

4. El conjunto de datos $evaluados2 - sinA7$ aplicando selección de atributos y eliminando los valores por encima de 2800 (870/900 instancias). Atributos de entrada: 24. (4)

5. El conjunto de datos $evaluados2 - sinA7$ eliminando los valores por encima de 2800 (840/870 instancias). (2)

6. El conjunto de datos $evaluados2 - sinA7$ aplicando selección de atributos y eliminando los valores por encima de 2800 (870/900 instancias). Atributos de entrada: 21. (4)

7. El conjunto de datos $evaluados2 - A7$. (5)

8. El conjunto de datos $evaluados2 - A7$ eliminando los valores por encima de 2800 (870/900 instancias) (6)

9. El conjunto de datos $evaluados2 - A7$ aplicando selección de atributos. Atributos de entrada:3, 4, 9, 16, 24, 25, 28 y 31. (7)

10. El conjunto de datos $evaluados2 - A7$ aplicando selección de atributos y eliminando los valores por encima de 2800 (870/900 instancias). Atributos de entrada: 15, 16, 25 y 31. (8)

11. El conjunto de datos $evaluados2 - A7, A5$. (9)

12. El conjunto de datos $evaluados2 - A7, A5$ eliminando los valores por encima de 2800 (870/900 instancias). (10)

13. El conjunto de datos $evaluados2 - A7, A5$ aplicando selección de atributos. Atributos de entrada: 3, 4, 9, 15, 16, 23, 24, 25,26, 28, 29 y 31. (11)

14. El conjunto de datos $evaluados2 - A7, A5$ aplicando selección de atributos y eliminando los valores por encima de 2800 (870/900 instancias). Atributos de entrada: 2, 3, 9, 12, 18, 24 y 26. (12)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	92,07(5,26)	96,97(7,55)	95,20(5,60)	95,17(5,37)	92,32(5,33)
(2)	100(0,0)	78,27(11,55)	82,15(12,28)	80,68(12,02)	81,45(11,98)	79,61(11,00)
(3)	100(0,0)	92,23(5,16)	91,45(5,84)	91,45(5,84)	91,45(5,84)	92,23(5,16)
(4)	100(0,0)	80,18(10,76)	77,62(11,84)	77,62(11,84)	77,62(11,84)	80,18(10,76)
(5)	100(0,0)	100,00(0,00)	103,73(1,23)	104,11(1,21)	103,91(1,08)	100,00(0,00)
(6)	100(0,0)	100,00(0,00)	103,03(0,89)	103,00(0,88)	103,01(0,88)	100,00(0,00)
(7)	100(0,0)	5,28(1,12)	42,47(15,90)	46,10(10,65)	47,60(8,54)	73,35(5,18)
(8)	100(0,0)	6,04(1,83)	35,16(16,75)	38,62(9,17)	42,64(8,06)	61,83(7,56)
(9)	100(0,0)	5,27(1,12)	**1,34(0,34)**	**1,34(0,34)**	**1,34(0,34)**	76,25(4,39)
(10)	100(0,0)	5,87(1,90)	6,07(9,49)	24,14(12,05)	28,41(9,54)	64,31(7,71)
(11)	100(0,0)	5,28(1,12)	36,23(15,72)	42,18(10,49)	47,60(8,22)	73,35(5,46)
(12)	100(0,0)	6,04(1,83)	29,91(17,49)	39,36(9,46)	44,34(7,78)	61,40(7,61)
(13)	100(0,0)	5,31(1,12)	4,63(5,85)	13,20(5,30)	14,99(3,35)	75,21(4,59)
(14)	100(0,0)	5,87(1,90)	3,32(4,78)	20,13(13,14)	27,03(9,45)	62,52(7,60)

Cuadro 4.20: Raíz cuadrada del error relativo medio y desviación típica de *storage* evaluados2

En esta prueba se han incluido más filtros porque después de aplicar un primer filtro de eliminación de valores fuera de rango, se comprobó que existían una pequeña cantidad que hacía empeorar la media del error.

Los mejores resultados de este experimento se dan con el fichero (9) con un valor de $1,34$. Este fichero posee como atributos de entrada la heurística del segundo problema ($heuristica2$), las metas del primer y segundo problema ($metas1$ y $metas2$), los literales del segundo problema ($literales2$), los objetos del segundo problema ($objetos2$), la distancia de los estados iniciales ($estadosiniciales$), la comparación del *domain transition graph* (dtg), los nodos evaluados del problema 1 ($evaluados1$) . Lo que quiere decir, que para conseguir el valor de $evaluados2$ se necesita tener información de los grafos.

4.10.3. Función objetivo: evaluadosc

Los conjuntos de datos que se utilizan para $evaluadosc - A7$, $evaluadosc - sinA7$ y $evaluadosc - A7, A5$ se encuentran divididos en:

1. El conjunto de datos $evaluadosc - sinA7$. (1)
2. El conjunto de datos $evaluadosc - sinA7$ eliminando los valores por encima de 12700 (893/900 instancias). (2)
3. El conjunto de datos $evaluadosc - sinA7$ aplicando selección de atributos. Atributos de entrada: 9, 16, 24, 25, 28 y 31. (3)
4. El conjunto de datos $evaluadosc - sinA7$ aplicando selección de atributos eliminando los valores por encima de 12700 (893/900 instancias).. Atributos de entrada: 6, 9, 16, 24, 25, 28 y 31. (4)
5. El conjunto de datos $evaluadosc - A7$. (1)
6. El conjunto de datos $evaluadosc - A7$ eliminando los valores por encima de 2200 (880/900 instancias). (2)

7. El conjunto de datos $evaluadosc - A7$ aplicando selección de atributos. Atributos de entrada: 3, 4, 9, 16, 23, 24, 25, 26, 28 ,29 y 31. (3)

8. El conjunto de datos $evaluadosc - A7$ aplicando selección de atributos eliminando los valores por encima de 12700 (893/900 instancias).. Atributos de entrada: 2, 3, 9, 12, 18, 24 y 25. (4)

9. El conjunto de datos $evaluadosc - A7, A5$. (1)

10. El conjunto de datos $evaluadosc - A7, A5$ eliminando los valores por encima de 12700 (893/900 instancias). (2)

11. El conjunto de datos $evaluadosc - A7, A5$ aplicando selección de atributos. Atributos de entrada: 3, 4, 9, 14, 16, 24, 25 y 31. (3)

12. El conjunto de datos $evaluadosc - A7, A5$ aplicando selección de atributos eliminando los valores por encima de 12700 (893/900 instancias).. Atributos de entrada: 2, 3, 9, 12, 18, 24 y 26. (4)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	29,06(25,72)	22,61(8,89)	38,23(12,76)	44,83(15,82)	51,59(15,64)
(2)	100(0,0)	30,72(39,33)	22,48(8,19)	30,94(7,91)	33,61(7,84)	63,91(9,05)
(3)	100(0,0)	19,21(9,70)	18,32(14,52)	30,16(13,11)	35,67(14,54)	52,09(15,69)
(4)	100(0,0)	20,17(8,30)	**15,40(6,73)**	20,52(7,69)	22,06(8,30)	64,52(8,71)
(5)	100(0,0)	20,29(13,01)	38,17(18,49)	47,81(12,65)	51,73(14,81)	42,98(13,09)
(6)	100(0,0)	19,89(6,57)	36,53(12,13)	37,68(9,02)	37,80(7,32)	54,69(6,85)
(7)	100(0,0)	16,96(7,63)	37,97(16,50)	45,25(12,50)	50,38(13,42)	49,87(14,52)
(8)	100(0,0)	62,55(77,07)	54,45(10,79)	54,14(9,04)	50,84(8,39)	58,63(7,35)
(9)	100(0,0)	21,32(14,74)	35,72(18,53)	49,27(15,06)	53,26(15,95)	43,05(13,07)
(10)	100(0,0)	21,03(10,07)	31,12(11,40)	34,88(8,80)	37,02(7,61)	55,01(6,97)
(11)	100(0,0)	18,46(10,20)	44,31(16,63)	47,10(14,66)	53,20(14,14)	47,43(13,97)
(12)	100(0,0)	49,13(10,55)	41,88(9,94)	39,94(8,53)	40,83(7,16)	59,96(6,67)

Cuadro 4.21: Raíz cuadrada del error relativo medio y desviación típica de *storage* evaluadosc

Los mejores resultados se encuentran en el fichero (4) tratándose de un fichero de datos que contiene un subconjunto de los datos de entrada ($A1$, $A2$ y $A3$) obteniendo un valor de $15,40$.

4.10.4. Función objetivo: longitud1

Los conjuntos de datos que se utilizan para *longitud*1 se encuentran divididos en:

1. El conjunto de datos *longitud*1 completo. Atributos de entrada $A1$. (1)

2. El conjunto de datos *longitud*1 aplicando selección de atributos. Atributos de entrada: 15 y 24. (3)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	25,34(12,08)	**9,21(8,54)**	11,38(8,76)	27,75(21,51)	71,23(9,54)
(2)	100(0,0)	52,77(10,45)	48,38(8,46)	50,68(9,04)	50,90(6,58)	71,81(9,45)

Cuadro 4.22: Raíz cuadrada del error relativo medio y desviación típica de *storage* longitud1

En esta prueba la que obtiene mejores resultados es el fichero (1) con un valor de $9,21$. Lo que quiere decir que necesita todos los datos de entrada $A1$, para conseguir un buen resultado de $longitud1$.

4.10.5. Función objetivo: longitud2

Los conjuntos de datos que se utilizan para $longitud2-sinA7$, $longitud2-A7$ y $longitud2-A7, A5$ se encuentran divididos en:

1. El conjunto de datos completo. Atributos de $longitud2-sinA7$. (1)
2. El conjunto $longitud2-sinA7$ aplicando selección de atributos. Atributo de entrada: 16, 24, 25, 28 y 31. (2)
3. El conjunto de datos completo. Atributos de$longitud2-A7$. (3)
4. El conjunto $longitud2-A7$ aplicando selección de atributos. Atributo de entrada: 9, 25, 28 y 31. (4)
5. El conjunto de datos completo. Atributos de $longitud2-A7, A5$. (5)
6. El conjunto $longitud2-A7, A5$ aplicando selección de atributos. Atributo de entrada: 16, 23, 25, 28 y 31. (6)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	12,35(6,90)	43,02(10,70)	50,66(6,56)	52,19(6,26)	56,26(4,37)
(2)	100(0,0)	11,06(1,50)	1,64(0,29)	1,64(0,29)	1,64(0,29)	56,26(4,37)
(3)	100(0,0)	50,97(50,98)	53,75(8,43)	49,47(5,68)	49,04(5,21)	61,42(5,27)
(4)	100(0,0)	44,40(4,38)	57,30(5,99)	52,56(5,54)	49,92(5,02)	61,16(5,48)
(5)	100(0,0)	11,07(1,43)	48,43(10,52)	50,05(6,49)	50,59(5,71)	56,09(4,45)
(6)	100(0,0)	11,10(1,48)	**1,62(0,29)**	**1,62(0,29)**	**1,62(0,29)**	56,26(4,37)

Cuadro 4.23: Raíz cuadrada del error relativo medio y desviación típica de Logistics longitud2

En este caso se produce prácticamente un empate en los mejores resultados entre los ficheros (2) y (6) (ligeramente mejor el 6). Esto se ha producido porque los dos conjuntos resultantes se les ha realizado un filtro de atributos y se han quedado con los mismos excepto 1. En el fichero (2) se han quedado con las metas del primer problema ($metas1$) y en el fichero (6) con el atributo derivado de metas ($metasc$). Como se puede deducir el resto se encuentran contenidos en los grupos de atributos $A1$, $A2$ y $A3$. Lo que quiere decir que no utilizan los datos relacionados con los grafos contenidos en el grupo $A7$ y la medida realizada no resulta útil.

4.10.6. Función objetivo: longitudc

Los conjuntos de datos que se utilizan para $longitudc-sinA7$, $longitudc-A7$ y $longitudc-A7, A5$ se encuentran divididos en:

1. El conjunto de datos completo. Atributos de $longitudc-sinA7$. (1)
2. El conjunto $longitudc-sinA7$ aplicando selección de atributos. Atributo de entrada: 6, 9, 12, 16, 18, 25, 28 y 31. (2)
3. El conjunto de datos completo. Atributos de $longitudc-A7$. (3)
4. El conjunto $longitudc-A7$ aplicando selección de atributos. Atributo de entrada: 2, 3, 9 y 12. (4)
5. El conjunto de datos completo. Atributos de $longitudc-A7, A5$. (5)
6. El conjunto $longitudc-A7, A5$ aplicando selección de atributos. Atributo de entrada: 4, 6, 14, 16 y 26. (6)

	ZeroR	M5Rules	IBK k = 1	IBK k =3	IBK k=5	linealRegression
(1)	100(0,0)	50,35(67,66)	44,93(11,69)	50,84(8,43)	53,11(8,22)	85,41(6,68)
(2)	100(0,0)	56,63(7,16)	58,22(8,31)	56,24(7,65)	55,30(7,38)	87,01(6,75)
(3)	100(0,0)	65,51(219,72)	51,12(10,83)	49,60(7,71)	50,73(7,16)	75,91(6,83)
(4)	100(0,0)	56,63(7,16)	58,22(8,31)	56,24(7,65)	55,30(7,38)	87,01(6,75)
(5)	100(0,0)	59,48(154,62)	48,81(10,76)	49,01(7,50)	51,24(7,33)	70,06(7,69)
(6)	100(0,0)	56,84(8,77)	**39,01(11,60)**	51,47(8,63)	53,45(8,01)	71,00(6,01)

Cuadro 4.24: Raíz cuadrada del error relativo medio y desviación típica de *Logistics* longitudc

Los mejores resultados se han obtenido son los del fichero (6), el error que ha obtenido es de $39,01$. Este fichero está compuesto de una selección de atributos de los conjuntos $A1$, $A2$, $A3$, $A5$ y $A7$. Esta selección se compone de la diferencia de los estados iniciales ($estadosiniciales$), de la longitud de la solución del problema 1 ($longitud1$), de la heurística del problema 2 ($heuristica2$) y del atributo derivado de literales ($literalesc$). La inclusión de los estados iniciales y no del resto de comparaciones del grupo $A7$, significa que los grafos son similares pero las situaciones iniciales no lo son, por eso es importante este peso.

4.10.7. Análisis de resultados

En este dominio, al igual que en el primero, tampoco se puede concluir que sea fundamental el uso de la comparación de los grafos para predecir los atributos de salida. Sin embargo, en los casos que se incluyen aparece el atributo relacionado con los estados iniciales, lo que significa que los grafos son parecidos, pero los estados iniciales distintos.

Los atributos más importantes son las heurísticas de los dos problemas ($heuristica1$ y $heuristica2$), la longitud del primer problema ($longitud1$) y objetos del primer problema ($objetos2$). En lo referente a las variables utilizadas del grupo $A7$, la más importante se trata de la diferencia de los estados iniciales ($estadosiniciales$) ya que al poseer grafos similares, la diferencia radica en ese atributo.

4.11. Comparación de los resultados

Como se ha podido ir viendo en la resolución de las distintas pruebas, la medida de distancia de los grafos no es al 100 % determinante para la predicción de los valores de salida. Sin embargo, en muchos casos mejora notablemente la solución de la misma. La eficacia de la inclusión de los atributos radica en la similitud de los mismos. Si estos son similares, y el valor de los estados iniciales difiere, entonces intervendrán; sin embargo, si los tres valores aportados son muy similares, ninguno de los tres formarán parte de la solución.

En el caso de que los grafos ofrecidos sean lo suficientemente dispares para ofrecer valores más amplios, todos sus atributos se utilizarán.

Esto lleva a pensar que la medida de similitud es acertada, ya que cuando los problemas son muy dispares ayuda a predecir los valores, y cuando los valores de los grafos son similares y los estados iniciales no lo son, este valor te ayuda a saber que el problema es distinto porque las situaciones iniciales son distintas. Esta información, sin la medida de similitud, no lo habríamos sabido a priori, sin mirar los dos grafos de los problemas.

En cuanto a los mejores resultados de los tres dominios se detallan en el último apéndice. En el cuadro aparecen los mejores resultados de los tres dominios. Estos resultados plasman la raíz cuadrada del error relativo y la desviación típica.

	Mejor resultado	Función objetivo
Logistics	41,92 (20,64)	evaluados1
	3,60 (1,28)	evaluados2
	27,65 (13,53)	evaluadosc
	2,42(0,51)	longitud1
	26,57 (2,87)	longitud2
	3,77 (0,38)	longitudc
Parking	46,85 (12,50)	evaluados1
	34,78 (17,77)	evaluados2
	31,86 (5,14)	evaluadosc
	17,02(2,49)	longitud1
	20,80 (3,21)	longitud2
	59,22 (20,46)	longitudc
Storage	19,18 (6,40)	evaluados1
	1,34(0,34)	evaluados2
	15,40 (6,73)	evaluadosc
	9,21 (8,54)	longitud1
	1,62 (0,29)	longitud2
	39,01 (11,60)	longitudc

Cuadro 4.25: Comparación resultados

Capítulo 5

Gestión del proyecto

En este capítulo se explican todas las tareas para el desarrollo del proyecto. Especifica las fases del proyecto, los medios empleados, la planificación del proyecto y el presupuesto de la realización del proyecto.

5.1. Fases del desarrollo

- Investigación sobre el estado del arte:
  - Planificación automática: En esta subfase se ha estudiado la diferencia entre los distintos planificadores y sus funcionamientos, además de las distintas formalizaciones.
  - Distancia entre grafos: En esta subfase se ha estudiado las distintas técnicas en función del tipo de grafo para poder establecer una comparación.
  - Métodos de análisis de datos: En esta subfase se ha estudiado las distintas técnicas para analizar los datos.
- Creación de conceptos únicos: Esta fase se centra en sacar los conceptos necesarios para formar la base del proyecto. Así cómo la realización de esquemas para plasmar su significado.
- Desarrollo de la aplicación: Durante esta fase se ha implementado el sistema para posteriormente realizar los experimentos.
- Experimentación: Esta fase corresponde con la elaboración de una serie de experimentos para probar el funcionamiento de la herramienta.
- Redacción de la memoria: Se realiza la fase de escritura del presente documento.
- Realización de la presentación: Se realiza la presentación y defensa del proyecto.

A continuación se muestra un diagrama en el tiempo con las distintas fases y el tiempo que ha ocupado cada una de ellas.

		Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	✓	**Proyecto Fin de Carrera**	**151 días?**	**vie 04/03/11**	**vie 30/09/11**	
2	✓	Reunión introducción proyecto	1 hora	vie 04/03/11	vie 04/03/11	
3	✓	Investigación	20 días?	lun 07/03/11	vie 01/04/11	2
4	✓	Creación de los conceptos básico	10 días?	lun 04/04/11	vie 15/04/11	3
5	✓	Desarrollo de la aplicación	46 días?	lun 18/04/11	lun 20/06/11	4
6	✓	Experimentación	30 días?	lun 20/06/11	vie 29/07/11	5
7	✓	**Memoria**	**39 días?**	**lun 01/08/11**	**jue 22/09/11**	**6**
8	✓	Contenido teorico	10 días?	lun 01/08/11	vie 12/08/11	
9	✓	Experimentación	10 días?	lun 15/08/11	vie 26/08/11	8
10	✓	Conclusiones	19 días?	lun 29/08/11	jue 22/09/11	9
11	✓	Reunión final proyecto	8 horas	vie 23/09/11	vie 23/09/11	10
12	✓	Presentación	1 día?	vie 30/09/11	vie 30/09/11	11

Figura 5.1: Diagrama de Gant (cuadro de tareas)

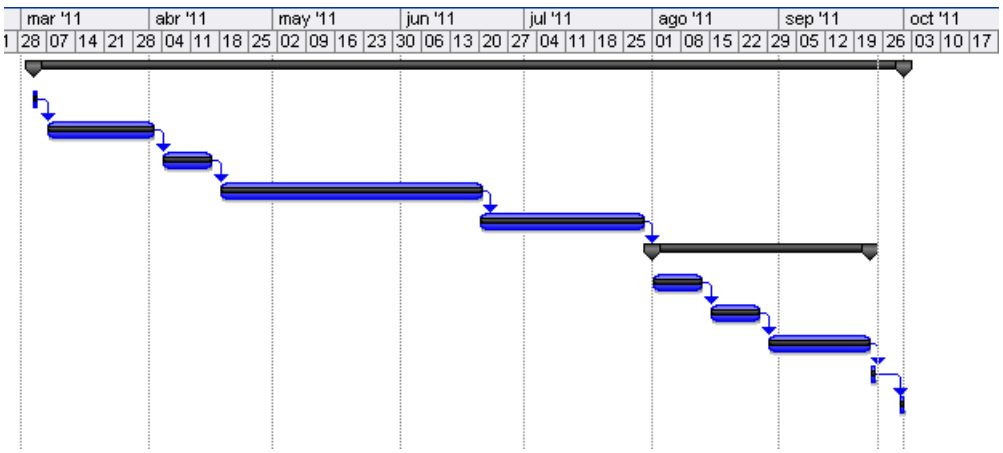


Figura 5.2: Diagrama de Gant (gráfico)

5.2. Medios empleados

El hardware necesario para realizar las pruebas experimentales y la documentación ha consistido en un ordenador de sobremesa, y un ordenador portátil, cuyas especificaciones detallamos a continuación.

- Ordenador de sobremesa:
  - Procesador: Pentium 4 @ 3,07 GHz.
  - Memoria RAM: 1 GB.
  - Sistema Operativo: Ubuntu 11.04.
- Ordenador portátil:
  - Procesador: Intel Core Duo @ 1,87 GHz.

- Memoria RAM: 4 GB.
- Sistema Operativo: Ubuntu 11.04 /Windows 7 Professional 32 bits.

En cuanto al acceso a redes, se ha usado una conexión ADSL (Orange 15 Mb) a Internet y la red VPN de la Universidad Carlos III, usada esta última para poder tener acceso libre a recursos académicos necesarios para la correcta documentación del proyecto.

La lista de software utilizado es: AdobeReader, Ubuntu, FireFox, lyx, latex, LAMA y Notepad++, Weka [WF05],LibreOffice 3.3.3, de libre distribución; Windows 7 y Microsof Office con licencia universitaria. Por último, indicar que se han usado servidores de correo gratuitos (GMail) para la comunicación y servidores gratuitos de almacenamiento de archivos (DropBox) para guardar y compartir el trabajo desarrollado.

5.3. Presupuesto

El presupuesto[1] es el siguiente:

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Isabel Rosario Cenamor Guijarro
2.- Departamento:
Escuela Politecnica superior, Ingeniería en Informica, Departamento de Informática
3.- Descripción del Proyecto:
- Titulo: Desarrollo y Selección de Características basadas en Distancias entre Grafos para Problemas de Predicción en Planificación Automática
- Duración (meses):
Tasa de costes Indirectos: 20%

4.- Presupuesto total del Proyecto (valores en Euros):
30.133 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación a) (hombres mes)	Coste hombre mes	Coste (Euro)
de la Rosa Turbides, Tomas		Ingeniero Senior	1	4.289,54	4.289,54
Fernández Rebollo, Fernando		Ingeniero Senior	1	4.289,54	4.289,54
Cenamor Guijarro, Isabel Rosario		Ingeniero	6	2.694,39	16.166,34
					0,00
					0,00
		Hombres mes 8		Total	24.745,42

a) 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

Figura 5.3: Presupuesto - parte I

[1]Plantilla de presupuesto: Fichero en excel con un modelo para ayudar a confeccionar el presupuesto del proyecto fin de carrera. plantilla [Junio de 2011]

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Pentium 4, 3,07 GHz	499,00	80	4	60	23,29
Intel Core Duo 1,87 Ghz	650,00	20	2	60	4,33
				60	0,00
				60	0,00
				60	0,00
					0,00
				Total	27,62

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Dietas	Restaurantes	200,00
Transporte	Cercanias Renfe	90,00
ADSL	Orange	108,00
Material de Oficina	Papelerias	20,00
Luz	Iberdrola	120,00
	Total	338,00

Figura 5.4: Presupuesto - parte II

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6.- Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	24.745
Amortización	28
Subcontratación de tareas	0
Costes de funcionamiento	338
Costes Indirectos	5.022
Total	30.133

Figura 5.5: Presupuesto - parte III

Capítulo 6

Conclusiones

En este proyecto se ha desarrollado un sistema que compara dos problemas en formulación SAS+ con el objetivo de extraer una serie de relaciones entre los distintos atributos. Para llevar a cabo esta tarea, se ha necesitado el apoyo de un planificador que utilizase esa formulación, como ha sido LAMA.

La inmensa mayoría de los problemas que se han utilizado en el sistema tardan un tiempo razonable en comparar todos los elementos. En este caso los problemas empleados no han sido de un tamaño elevado, no han existido problemas que tengan más de 70 objetos, que corresponden a un máximo de 64 variables en SAS+. Siendo un total de 90 problemas de 3 dominios diferentes.

Los problemas de gran tamaño, tal y como esta implementado el sistema, supondrían un coste demasiado elevado, debido a la complejidad que poseen los métodos. Con lo que se incluye una muestra representativa de problemas, que no posean un tamaño que condicione los resultados.

A pesar de que no se ha realizado en más dominios, los resultados obtenidos son bastante buenos. Se han obtenido valores en la raíz cuadrada del error relativo inferiores al 50 % para el conjunto de los 90 problemas.

Una de las principales conclusiones a la que puede llegarse tras la implementación de este sistema independiente de dominio, es que no solo puedes comparar problemas del mismo dominio; sino que puedes emplearlo con toda clase de problemas. Esto podría desembocar en la predicción del tiempo de ejecución de otros sistemas más complejos con la simple comparación de los atributos proporcionados por los distintos problemas.

Como se ha observado, se han obtenido mejores resultados teniendo en cuenta todos los atributos del sistema, que realizando una selección de los mismos. Como se ha visto, los atributos más importantes son los relacionados con las heurísticas, las metas, los literales y los objetos, además de los relacionados con los grafos, el *domain transition graph*, el *causal graph* y los estados iniciales. Además tratándose de distintos problemas, esta cuestión no ha resultado unánime, por el hecho que en uno de los dominios, los resultados mejoraron al ofrecer una selección de los atributos. Sin embargo, a nivel general, se ha observado que todos los problemas conjuntamente mejoran con todos los atributos.

Una parte negativa de la utilización de todos los atributos, es que no se puede realizar la clasificación con los atributos conocidos a priori; es decir, los que se conocen antes de obtener la solución. Esto conlleva, en el caso de que se quiera llegar a predecir la solución, estimar esos valores. Sin embargo, también

aporta la certeza de que los resultados sean más acertados, porque la solución siempre esta relacionada con los atributos a posteriori; que son todos aquellos que se obtienen una vez se ha conseguido la solución.

La parte positiva del resultado obtenido, es que además de evaluar el resultado de la fórmula propuesta para la comparación de los grafos, se han obtenido el conjunto de variables que hay que tener en cuenta para la predicción de los atributos de salida.

En resumen, puede concluirse que la medida aportada de comparación de los grafos de los problemas es útil para predecir los valores de salida, ya que con su utilización mejoran los resultados. Pero es probable que exista otra fórmula que ayude mejor a la clasificación que la propuesta en este trabajo, ya que en problemas con grafos muy similares, los resultados obtenidos no incorporan los valores de estos atributos. Y sería muy útil, que con esas variables ayudase a reducir el error.

Capítulo 7

Trabajos futuros

Una vez finalizado el desarrollo del proyecto y tras extraer una serie de conclusiones posteriores a la evaluación del mismo, pueden plantearse ciertas mejoras o líneas de trabajo para complementar las funcionalidades del mismo. A continuación se describen una serie de propuestas en este sentido:

- Inclusión de heurísticas para el cálculo de isomorfismo.

  Debido a la utilización de fuerza bruta en la búsqueda de las matrices isomorfas, ha supuesto una limitación computacional en el desarrollo del sistema. Si se reduce la complejidad de esos métodos se podrá emplear el programa en problemas más grandes.

- Inclusión de heurísticas para la comparación de grafos.

  Debido a los subniveles de los grafos y sobre todo a que sus aristas no están codificadas numéricamente, la comparación textual de los mismos es muy elaborada. Para aumentar el rendimiento en este caso, habría que buscar una representación numérica que pueda representar semánticamente lo mismo que de la manera actual.

- Planificador basado en instancias.

  Una vez que se ha obtenido la recta de clasificación, una ampliación del sistema sería la creación de un planificador basado en instancias. Este planificador obtendría la solución a partir de la función obtenida, y daría la solución del problema que menor valor obtuviera. De esta manera, se ahorraría el coste computacional que conlleva calcular la solución.

- Tipificación de las variables SAS+.

  Para aumentar el rendimiento, o mejorar el entendimiento del sistema, dividir las variables proporcionadas por tipos. Esto quiere decir, por ejemplo en el caso del dominio *logistics*, decir qué variables corresponden a paquetes, a aviones, etc. Esto simplificaría la comparación entre grafos, porque se podrían comparar con los que perteneciesen al mismo tipo. El problema que esto conllevaría es que tiene que realizarse nada más cargarse el problema y no pueden aparecer tipos nuevos posteriormente, tendrían que ser fijados a priori.

Glosario

- Acción u Operador: transformación de un estado en otro.
- CG: Causal Graph
- DTG: Domain Transition Graph
- Eff: Efectos (consecuentes)
- Estado o situación: descripción instantánea del entorno.
- Estado inicial: situación de partida.
- FF: Fast-Forward.
- Grafo: Se trata de un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.
- GRT: Un planificador de espacio de estado con búsqueda hacia delante.
- Heurística: Se trata de un método basado en la experiencia que puede utilizarse como ayuda para resolver problemas de diseño.
- HSP: Heuristic Search Planner.
- Objetivo o meta: descripción de condiciones que se tienen que dar para considerar por terminado el proceso.
- PDDL: Planning Domain Definition Language.
- Plan: secuencia de operadores que permiten pasar del estado inicial a un estado en el que se cumplan los objetivos.
- Pre: Precondiciones (restricciones).
- SAS+: Heurística basada en grafos causales.
- STRIPS: Stanford Research Institute Problem Solver.

Bibliografía

[AKA91] D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine learning*, 6, 1991.

[Bac92] C. Backstrom. Equivalence and tractability results for sas+ planning. In *Proceedings of the 3rd International Conference on Principles on Knowledge Representation and Reasoning (KR-92*. Cites, 1992.

[Bac01] F. Bacchus. Aips 2000 planning competition: The fifth international conference on artificial intelligence planning and scheduling systems. *Ai magazine*, 2001.

[BC96] G. Briscoe and T. Caelli. *A Compendium of Machine Learning: Symbolic Machine Learning*, volume 1. Ablex Pub, 1996.

[BF97] A.L. Blum and M.L. Furst. Fast planning through planning graph analysis* 1. *Artificial intelligence*, 90(1-2):281–300, 1997.

[BG98] B. Bonnet and H. Geffner. Hsp: Heuristic search planner. 1998.

[BK00] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2):123–191, 2000.

[Blu95] A.L. Blum. Fast planning through planning graph analysis. Technical report, DTIC Document, 1995.

[BM09] J.A. Baier and S.A. McIlraith. Planning with preferences. *AI Magazine*, 29(4):25, 2009.

[BN93] C. Backstrom and B. Nebel. Complexity results for sas+ planning. In *Computational Intelligence*. Citeseer, 1993.

[BS98] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern recognition letters*, 1998.

[BV97] J. Blythe and M. Veloso. Analogical replay for efficient conditional planning. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 668–673. Citeseer, 1997.

[Byl94] T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

[Cen87] J. Cendrowska. Prism: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349–370, 1987.

[CR95] Y. Chauvin and D.E. Rumelhart. *Backpropagation: theory, architectures, and applications*. Lawrence Erlbaum, 1995.

[DB02] C. Domshlak and R.I. Brafman. Structure and complexity in planning with unary operators. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling*, 2002.

[DD01] C. Domshlak and Y. Dinitz. Multi-agent off-line coordination: Structure and complexity. In *Proceedings of the 6th European Conference on Planning*. Citeseer, 2001.

[DH73] R.O. Duda and P.E. Hart. Pattern classification and scene analysis. *A Wiley-Interscience Publication, New York: Wiley, 1973*, 1, 1973.

[DK03] M.B. Do and S. Kambhampati. Sapa: A multi-objective metric temporal planner. *Journal of Artificial Intelligence Research*, 20(1):155–194, 2003.

[ERMC00] T. Estlin, G. Rabideau, D. Mutz, and S. Chien. Using continuous planning techniques to coordinate multiple rovers. *Electronic Transactions on Artificial Intelligence*, 4:45–57, 2000.

[Fis87] D. Fisher. Improving inference through conceptual clustering. In *Proc. 1987 AAAI Conf*, pages 461–465, 1987.

[FL03] M. Fox and D. Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20(1):61–124, 2003.

[FN72] R.E. Fikes and N.J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1972.

[Fog94] D.B. Fogel. An introduction to simulated evolutionary optimization. *Neural Networks, IEEE Transactions on*, 5(1):3–14, 1994.

[GFG+97] C. Ghez, M. Favilla, MF Ghilardi, J. Gordon, R. Bermejo, and S. Pullman. Discrete and continuous planning of hand movements and isometric force trajectories. *Experimental Brain Research*, 115(2):217–233, 1997.

[GHK+98] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl the planning domain definition language. *AIPS-98 planning committee*, 1998.

[Gol89] D.E. Goldberg. Genetic algorithms in search, optimization, and machine learning. 1989.

[GS02] A. Gerevini and I. Serina. Lpg: A planner based on local search for planning graphs with action costs. In *Proc. of the Sixth Int. Conf. on AI Planning and Scheduling*, pages 12–22, 2002.

[GSS04] A. Gerevini, A. Saetti, and I. Serina. Planning with numerical expressions in lpg. 2004.

[Hel04] M. Helmert. A planning heuristic based on causal graph analysis. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 161–170, 2004.

[HHP99] G. Holmes, M. Hall, and E. Prank. Generating rule sets from model trees. *Advanced Topics in Artificial Intelligence*, pages 1–12, 1999.

[HN01] J. Hoffmann and B. Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14(1):253–302, 2001.

[Hof03] J. Hoffmann. The metric-ff planning system: Translating ignoring delete lists to numeric state variables. *Journal of Artificial Intelligence Research*, 20(1):291–341, 2003.

[Hol93] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90, 1993.

[Kam06] T. Kampke. Distance patterns in structural similarity. *The Journal of Machine Learning Research*, 2006.

[Knu77] D.E. Knuth. A generalization of dijkstra's algorithm* 1. *Information Processing Letters*, 6(1), 1977.

[LB03] A. Lopez and F. Bacchus. Generalizing graphplan by formulating planning as a csp. In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*. Citeseer, 2003.

[LRN86] J.E. Laird, P.S. Rosenbloom, and A. Newell. Chunking in soar: The anatomy of a general learning mechanism. *Machine Learning*, 1(1):11–46, 1986.

[M+67] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1. California, USA, 1967.

[MM92] Z. Michalewicz and Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*, volume 19. Springer-verlag Berlin, 1992.

[MPVV01] D.C. Montgomery, E.A. Peck, G.G. Vining, and J. Vining. *Introduction to linear regression analysis*. Wiley New York, 2001.

[MS83] R.S. Michalski and R.E. Stepp. Learning from observation: Conceptual clustering. *Machine Learning: An artificial intelligence approach*, 1983.

[NGT04] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers, 2004.

[NNK00] R.S. Nigenda, X.L. Nguyen, and S. Kambhampati. Altalt: Combining the advantages of graphplan and heuristic state search. In *International conference on knowledge-based computer systems*. Citeseer, 2000.

[PS92] M.A. Peot and D.E. Smith. Conditional nonlinear planning. In *Artificial intelligence planning systems: proceedings of the first international conference, June 15-17, 1992, College Park, Maryland*, page 189. Morgan Kaufmann Pub, 1992.

[PW92] J.S. Penberthy and D. Weld. Ucpop: A sound, complete, partial order planner for adl. In *proceedings of the third international conference on knowledge representation and reasoning*, pages 103–114. Citeseer, 1992.

[Qui86] J.R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[Qui87] J.R. Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[Qui93] J.R. Quinlan. *C4. 5: programs for machine learning*. Morgan Kaufmann, 1993.

[RBV01] I. Refanidis, N. Bassiliades, and I. Vlahavas. Ai planning for transportation logistics. In *Proceedings 17th International Logistics Conference*. Citeseer, 2001.

[Ric10] S. Richter. *Landmark-Based Heuristics and Search Control for Automated Planning*. PhD thesis, Griffith University, Brisbane, Australia, 2010.

[RMDUoCS86] D.E. Rumelhart, J.L. McClelland, and PDP Rese Diego) University of California (San. Parallel distributed processing. 1986.

[RV01] I. Refanidis and I. Vlahavas. The grt planner. *AI Magazine*, 22(3):63, 2001.

[RV11] I. Refanidis and I. Vlahavas. The grt planning system: Backward heuristic construction in forward state-space planning. *Arxiv preprint arXiv:1106.0285*, 2011.

[RW08] S. Richter and M. Westphal. The lama planner using landmark counting in heuristic search. In *Proceedings of IPC*, 2008.

[RW10] S. Richter and M. Westphal. The lama planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, 2010.

[SM96] D. Sloane and S.P. Morgan. An introduction to categorical data analysis. *Annual review of sociology*, pages 351–375, 1996.

[SP06] T.C. Son and E. Pontelli. Planning with preferences using logic programming. *Theory and Practice of Logic Programming*, 6(05):559–607, 2006.

[TTLY98] C.C. Track, C.P. Track, M. Littman, and H. Younes. International planning competition. 1998.

[VCP+95] M. Veloso, J. Carbonell, A. Perez, D. Borrajo, E. Fink, and J. Blythe. Integrating planning and learning: The prodigy architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:81–81, 1995.

[W+01] D.B. West et al. *Introduction to graph theory*, volume 1. Prentice Hall Upper Saddle River, NJ, 2001.

[WF05] I.H. Witten and E. Frank. Data mining: Practical machine learing tools and and techniques. 2005.

[WFT+99] I.H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S.J. Cunningham. Weka: Practical machine learning tools and techniques with java implementations. In *ICONIP/ANZIIS/ANNES*, volume 99, pages 192–196. Citeseer, 1999.

[WK91] S.M. Weiss and C.A. Kulikowski. *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. M. Kaufmann Publishers San Mateo, Calif, 1991.

[WMMY02] R.E. Walpole, R.H. Myers, S.L. Myers, and K. Ye. Probability and statistics for engineers and scientists. 2002.

[WW96] Y. Wang and I.H. Witten. Induction of model trees for predicting continuous classes. 1996.

[Zad65] L.A. Zadeh. Fuzzy sets*. *Information and control*, 8(3):338–353, 1965.

Apéndice A

Ficheros de problemas

En este apéndice se describen el fichero de problema sin resolver, resuelto y del conjunto de datos.

A.1. Fichero problema

logistics
DTG
4 0 1 0 0
at(truck0, loc0-0)
at(truck1, loc1-1)
at(airplane0, loc0-0)
at(p0, loc1-0)
var:2 at(airplane0, loc0-0)+ at(airplane0, loc1-0)+fly-airplane
var:0 at(truck0, loc0-0)+ at(truck0, loc0-1)+drive-truck
var:1 at(truck1, loc1-1)+ at(truck1, loc1-0)+drive-truck
var:3 at(p0, loc1-0)+ in(p0, airplane0)+load-airplane
var:2 at(airplane0, loc1-0)+ at(airplane0, loc0-0)+fly-airplane
var:0 at(truck0, loc0-1)+ at(truck0, loc0-0)+drive-truck
var:3 at(p0, loc1-0)+ in(p0, truck1)+load-truck
var:1 at(truck1, loc1-0)+ at(truck1, loc1-1)+drive-truck
var:3 in(p0, airplane0)+ at(p0, loc0-0)+unload-airplane
var:3 in(p0, airplane0)+ at(p0, loc1-0)+unload-airplane
var:3 in(p0, truck1)+ at(p0, loc1-1)+unload-truck
var:3 in(p0, truck1)+ at(p0, loc1-0)+unload-truck
var:3 at(p0, loc0-0)+ in(p0, truck0)+load-truck
var:3 at(p0, loc0-0)+ in(p0, airplane0)+load-airplane
var:3 at(p0, loc1-1)+ in(p0, truck1)+load-truck
var:3 in(p0, truck0)+ at(p0, loc0-0)+unload-truck
var:3 in(p0, truck0)+ at(p0, loc0-1)+unload-truck
var:3 at(p0, loc0-1)+ in(p0, truck0)+load-truck
CG
var:0 - var:3,4
var:1 - var:3,4
var:2 - var:3,4

var:3

end

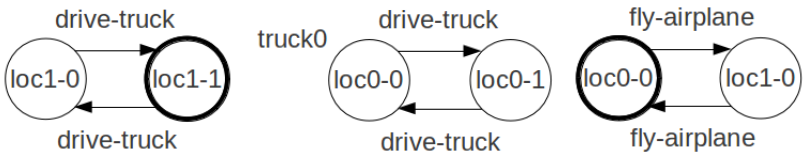
De esta salida surgen estos DTG:

- La variable 0 corresponde con el camión *truck*0 ilustrado en la sub-figura A.1a, donde se puede marcar el estado inicial en negro.

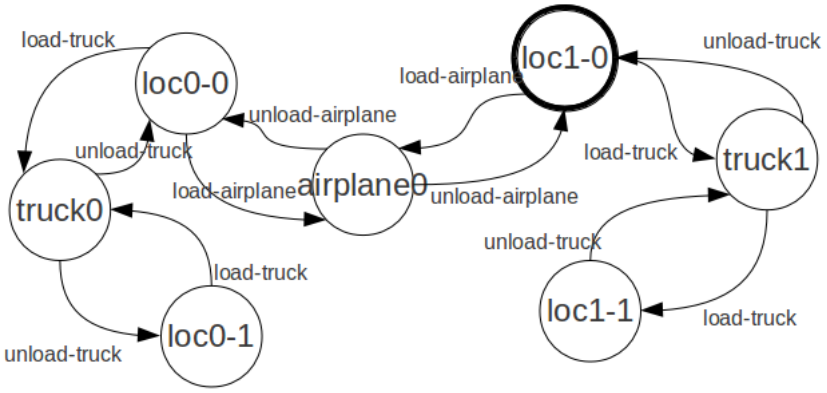
- La variable 1 corresponde con el camión *truck*1 ilustrado en la sub-figura A.1b, donde se puede marcar el estado inicial en negro.

- La variable 2 corresponde con el avión *airplane*0 ilustrado en la sub-figura A.1c, donde se puede marcar el estado inicial en negro.

- La variable 3 corresponde con el paquete *p*0 ilustrado en la sub-figura A.1d, donde se puede marcar el estado inicial en negro.



(a) Camión 0 (b) Camión 1 (c) Avión 0



(d) Paquete 0

Figura A.1: Grafos DTG problema *logistics*

Y el siguiente CG.

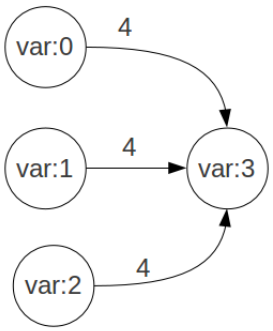


Figura A.2: CG problema *logistics*

A.2. Fichero de entrada solución

H(i): 3
H(i): 3
H(i): 2
H(i): 2
H(i): 1
H(i): 1
result: 1
(fly-airplane airplane0 loc0-0 loc1-0)
(load-airplane p0 airplane0 loc1-0)
(fly-airplane airplane0 loc1-0 loc0-0)
(unload-airplane p0 airplane0 loc0-0)
f
Plan length: 4 step(s).
Expanded 5 state(s).
Generated 15 state(s).
Search time: 0 seconds
Total time: 0 seconds
Objetos: 9
Literales: 7
Metas: 1
end

En la parte de la solución se puede ver en primer lugar los valores de las heurísticas que se han conseguido con el problema $H(i) : 3$. Después la solución del problema, es decir, su plan. A continuación vienen una serie de datos del problema: la longitud del plan, los nodos evaluados, los nodos generados, el tiempo de búsqueda, el tiempo total, el número de objetos, el número de literales y el número de metas.

A.3. Comparación con n problemas

Consiste en una concatenación de 1 a n problemas separados por la secuencia de separación "FFFFFF". Cada problema está compuesto por A.1 sin el "end" del final seguido por el apartado A.2.

Apéndice B

Fichero de salida y de entrada de Weka

En este apéndice se describen el fichero de salida del programa implementado y el de entrada de Weka transformado a partir del anterior.

B.1. Salida programa

DDD – Cargar Datos Resueltos: /home/isabel/PFC/pruebasLogistics/pruebasLogistics1/01-001
DDD– salimos del DTG del problema
DDD– salimos del CG del problema
DDD– salimos de problema a resolver
DDD – Cargar Datos Resueltos: /home/isabel/PFC/pruebasLogistics/todosLogistics
DDD — problema a cargar 0
DDD– salimos del DTG del problema
DDD – Entramos en las soluciones
DDD – Cargar Solucion
DDD —Intrododucimos un problema
DDD — problema a cargar 1
DDD– salimos del DTG del problema
DDD – Entramos en las soluciones
DDD – Cargar Solucion
DDD — Salimos de los problemas resueltos
DDD – Los problemas ya estan cargados
70DDD – Comparacion con el problema: 0
DDD – Valores del CG pasados
0 [0 0 0 4]
1 [0 0 0 4]
2 [0 0 0 4]
3 [0 0 0 0]
****
0 [0 0 0 4]
1 [0 0 0 4]
2 [0 0 0 4]

3 [0 0 0 0]
****
Calculando...
******* Los grafos son isomorfos
******* El valor de simetria de estos grafos es de: 0
DDD – Comparacion con el problema: 1
DDD – Valores del CG pasados
0 [0 0 0 4]
1 [0 0 0 4]
2 [0 0 0 4]
3 [0 0 0 0]
****
0 [0 0 0 4]
1 [0 0 0 4]
2 [0 0 0 4]
3 [0 0 0 0]
****
Calculando...
******* Los grafos son isomorfos
******* El valor de simetria de estos grafos es de: 0
DDD – Comparacion con el problema: 0
DDD – Inicio de comparacion DTG
******* Los problemas introducidos tienen una simetria de: 0
DDD – Distancia estados iniciales
DDD — Comparacion de los estados iniciales del problema
La diferencia de estados iniciales es de: 0
DDD – Comparacion con el problema: 1
DDD – Inicio de comparacion DTG
******* Los problemas introducidos tienen una simetria de: 0
DDD – Distancia estados iniciales
DDD — Comparacion de los estados iniciales del problema
La diferencia de estados iniciales es de: 7
DDD – Cargar Solucion /home/isabel/PFC/pruebasLogistics/pruebasLogistics1/sol001
DDD — Solución del problema a resolver
****
Problema: 0 s 0
Comparacion CG: 0
Comparacion DTG: 0
Comparacion estados: 0
Comparacion Longitud 0
Longitud p1: 4 Longitud p2: 4
Comparacion estados evaluados 0
P1: 5 P2: 5
Comparacion estados generados 0
P1: 15 P2: 15
Comparacion con H(i) 0
P1: 3 P2: 3
Comparacion t Busqueda 0
P1: 0 P2: 0
Comparacion t Total 0

P1: 0 P2: 0
Comparacion metas 0
P1: 1 P2: 1
Comparacion literales 0
P1: 8 P2: 8
Comparacion objetos 0
P1: 10 P2: 10
****
Problema: 1 s 0
Comparacion CG: 0
Comparacion DTG: 0
Comparacion estados: 7
Comparacion Longitud 6
Longitud p1: 10 Longitud p2: 4
Comparacion estados evaluados 6
P1: 11 P2: 5
Comparacion estados generados 27
P1: 42 P2: 15
Comparacion con H(i) 6
P1: 9 P2: 3
Comparacion t Busqueda 0
P1: 0 P2: 0
Comparacion t Total 0
P1: 0 P2: 0
Comparacion metas 0
P1: 1 P2: 1
Comparacion literales 0
P1: 8 P2: 8
Comparacion objetos 0
P1: 10 P2: 10
DDD – Fin del programa

B.2. Fichero entrada weka

@relation logistics1
@attribute numeroProblema real
@attribute cg real
@attribute dtg real
@attribute estadosIniciales real
@attribute longitudc real
@attribute longitud1 real
@attribute longitud2 real
@attribute evaluadosc real
@attribute evaluados1 real
@attribute evaluados2 real
@attribute generadosc real
@attribute generados1 real
@attribute generados2 real
@attribute heuristicac real

@attribute heuristica1 real
@attribute heuristica2 real
@attribute tBusquedac real
@attribute tBusqueda1 real
@attribute tBusqueda2 real
@attribute tTotalc real
@attribute tTotal1 real
@attribute tTotal2 real
@attribute metasc real
@attribute metas1 real
@attribute metas2 real
@attribute literalesc real
@attribute literales1 real
@attribute literales2 real
@attribute objetosc real
@attribute objetos1 real
@attribute objetos2 real
@data
20,0,0,0,0,48,48,0,130,130,0,2322,2322,0,38,38,0,0.02,0.02,0,0.03,0.03,0,10,10,0,21,21,0,24,24
21,0,0,32,10,58,48,61,191,130,556,2878,2322,5,43,38,0.01,0.03,0.02,0.01,0.04,0.03,0,10,10,0,21
,21,0,24,24
22,0,0,39,23,25,48,103,27,130,1943,379,2322,15,23,38,-0.02,0,0.02,-0.02,0.01,0.03,0,10,10,0,21
,21,0,24,24
23,0,0,29,5,43,48,52,78,130,954,1368,2322,2,36,38,-0.01,0.01,0.02,-0.01,0.02,0.03,0,10,10,0,21
,21,0,24,24
24,0,0,34,9,39,48,6,124,130,240,2082,2322,6,32,38,-0.01,0.01,0.02,-0.01,0.02,0.03,0,10,10,0,21
,21,0,24,24
25,320,150,45,1,47,48,11,141,130,330,2652,2322,1,39,38,0,0.02,0.02,0.01,0.04,0.03,0,10,10,6,
27,21,6,30,24
26,320,150,75,10,58,48,135,265,130,2340,4662,2322,7,45,38,0.02,0.04,0.02,0.03,0.06,0.03,0
,10,10,6,27,21,6,30,24
27,320,150,51,16,64,48,165,295,130,3050,5372,2322,13,51,38,0.03,0.05,0.02,0.04,0.07,0.03
,0,10,10,6,27,21,6,30,24
28,320,150,50,10,58,48,95,225,130,2502,4824,2322,9,47,38,0.02,0.04,0.02,0.03,0.06,0.03,0
,10,10,6,27,21,6,30,24
29,320,150,69,10,38,48,47,83,130,697,1625,2322,5,33,38,0,0.02,0.02,0,0.03,0.03,0,10,10,6
,27,21,6,30,24
30,1640,210,96,72,120,48,1393,1523,130,30749,33071,2322,61,99,38,0.39,0.41,0.02,0.45
,0.48,0.03,5,15,10,11,32,21,11,35,24
31,1640,210,76,29,77,48,172,302,130,4204,6526,2322,22,60,38,0.04,0.06,0.02,0.06,0.09
,0.03,5,15,10,11,32,21,11,35,24
32,1640,210,76,27,75,48,63,193,130,2137,4459,2322,29,67,38,0.02,0.04,0.02,0.05,0.08,0.03
,5,15,10,11,32,21,11,35,24
33,1640,210,68,15,63,48,1030,1160,130,461,2783,2322,19,57,38,0,0.02,0.02,0.03,0.06,0.03
,5,15,10,11,32,21,11,35,24
34,1640,210,81,43,91,48,221,351,130,5578,7900,2322,36,74,38,0.06,0.08,0.02,0.09,0.12
,0.03,5,15,10,11,32,21,11,35,24
35,15296,792,173,45,93,48,281,411,130,13462,15784,2322,36,74,38,0.11,0.13,0.02,0.17
,0.2,0.03,10,20,10,20,41,21,23,47,24
36,15296,792,207,38,86,48,102,232,130,6930,9252,2322,36,74,38,0.05,0.07,0.02,0.11,0.14

,0.03,10,20,10,20,41,21,23,47,24
37,15296,792,193,88,136,48,1060,1190,130,43906,46228,2322,59,97,38,0.42,0.44,0.02,0.49
,0.52,0.03,10,20,10,20,41,21,23,47,24
38,15296,792,216,64,112,48,543,673,130,21765,24087,2322,50,88,38,0.22,0.24,0.02,0.28,0.31
,0.03,10,20,10,20,41,21,23,47,24
39,15296,792,204,45,93,48,463,593,130,20852,23174,2322,36,74,38,0.17,0.19,0.02,0.22,0.25
,0.03,10,20,10,20,41,21,23,47,24
40,22592,1332,404,80,128,48,686,816,130,33545,35867,2322,64,102,38,0.32,0.34,0.02,0.42
,0.45,0.03,10,20,10,32,53,21,35,59,24
41,22592,1332,352,105,153,48,1070,1200,130,52046,54368,2322,84,122,38,0.55,0.57,0.02
,0.67,0.7,0.03,10,20,10,32,53,21,35,59,24
42,22592,1332,301,60,108,48,449,579,130,25314,27636,2322,49,87,38,0.22,0.24,0.02,0.31
,0.34,0.03,10,20,10,32,53,21,35,59,24
43,22592,1332,305,78,1260,48,688,818,130,36539,38861,2322,66,104,38,0.31,0.33,0.02,0.42
,0.45,0.03,10,20,10,32,53,21,35,59,24
44,22592,1332,354,82,130,48,684,814,130,34778,37100,2322,61,99,38,0.32,0.34,0.02,0.42
,0.45,0.03,10,20,10,32,53,21,35,59,24
45,26992,1632,342,121,169,48,1336,1466,130,66434,68756,2322,98,136,38,0.77,0.79,0.02
,0.92,0.95,0.03,15,25,10,37,58,21,40,64,24
46,26992,1632,359,95,143,48,1050,1180,130,55153,57475,2322,77,115,38,0.55,0.57,0.02,0.69
,0.72,0.03,15,25,10,37,58,21,40,64,24
47,26992,1632,346,109,157,48,1031,1161,130,53798,56120,2322,94,132,38,0.6,0.62,0.02,0.76
,0.79,0.03,15,25,10,37,58,21,40,64,24
48,26992,1632,440,124,172,48,1888,2018,130,91048,93370,2322,95,133,38,1.11,1.13,0.02,1.25
,1.28,0.03,15,25,10,37,58,21,40,64,24
49,26992,1632,432,109,157,48,1158,1288,130,57428,59750,2322,86,124,38,0.68,0.7,0.02,0.82
,0.85,0.03,15,25,10,37,58,21,40,64,24

Apéndice C

Manual de usuario

En este apéndice se describe como compilar el programa, como ejecutarlo y una explicación genérica del fichero de salida.

C.1. Compilación de clases

Para proceder a la compilación del programa de manera automática se creó un fichero tipo *Makefile*, el cual, se encuentra representado en el Cuadro C.1.

```
./make
g++ -ansi -Wall -O3 -Wno-sign-compare -pedantic -Werror arco.h
cg.h transicion.h variable.h problema.h main.cc dtg.cc arco.cc
cg.cc transicion.cc variable.cc problema.cc cabecera.h
MetodosComparacionCG.cc MetodosComparacionCG.h
metodoscomparaciondtg.h metodoscomparaciondtg.cc
comparacionCG.cc seleccionmejores.h seleccionmejores.cc
comparacion.h auxiliares.cc -o pfc -g
```

Cuadro C.1: Compilación del programa

C.2. Ejecución del problema

Una vez creado el ejecutable, se puede ejecutar el programa usando varias opciones para que funcione de manera correcta. Estas opciones están descritas en el Cuadro .

./pfc <ficheroResueltos> <problemaAResolver> <resultado>

Cuadro C.2: Linea de ejecución

Ser realiza una comparación entre todos los problemas que se encuentran en ficheroResueltos con el problema, así mismo la solución del problema se compara con el resto y se muestra por pantalla..

En la siguiente lista se explican con más detalle cada uno de los parámetros que han aparecido en las lineas de comandos:

./pfc: Nombre del programa que vamos a ejecutar.

ficheroResueltos: Se trata del fichero de problemas resueltos con el formato detallado en el capítulo 4, apartado obtención de los ficheros de datos. En este fichero pueden ir 1 a n problemas.

problemaAResolver: Se trata del fichero del problema. Solamente puede ir 1 problema siguiendo el formato descrito en el capítulo 4, apartado obtención de los ficheros de datos.

resultado: Fichero de texto con la solución del problema a resolver. Este parámetro es opcional. El formato que tiene que seguir es igual que es el de las soluciones de cualquier problema.

C.3. Salida del programa

Por pantalla se muestran la serie de pasos que siguen para conseguir la comparación del problema.

1. Lectura de los ficheros.
2. Por cada problema resuelto:
   a) Calculo de la matriz isomorfa para CG.
   b) Cálculo del coeficiente de comparación.
   c) Almacenar los resultados.
3. Por cada problema resuelto:
   a) Comprobación isomorfismo en DTG.
   b) Cálculo del coeficiente de comparación.
   c) Almacenar los resultados.
4. Si se encuentra la opción activada de estados.
   a) Cálculo la distancia entre estados.
   b) Almacenar los resultados.
5. Si se encuentra la opción activada de fichero resultados.
   a) Cálculo la distancia entre estados.
   b) Almacenar resultado.
6. Devolver mejor resultado obtenido.

Para ver más en detalle la salida del programa consultar en Anexo 2 sección 9.1.

Apéndice D

Tablas resumen por función de evaluación

En este apéndice se muestran los resultados por función evaluación.

Evaluados1

	Valor	Algoritmo	Fichero
Logistics	41.92(20.64)	IBK k=1,3,5	evaluados1 logistic - (2)
Parking	46,85(12,50)	IBK k=1,3,5	evaluados1 parking - (1)
Storage	19,18(6,40)	IBK k=1,3,5	evaluados1 storage - (4)

Cuadro D.1: Comparativa evaluados1

Evaluados2

	Valor	Algoritmo	Fichero
Logistics	3,60(1,28)	IBK k=1	evaluados2 logistic - (7), (11)
Parking	36,26(15,88)	M5Rules	evaluados2 parking - (2)
Storage	1,34(0,34)	IBK k=1,3,5	evaluados2 storage - (9)

Cuadro D.2: Comparativa evaluados2

Evaluadosc

	Valor	Algoritmo	Fichero
Logistics	31,56(16,33)	M5Rules	evaluadosc logistic - (15)
Parking	31,86(5,14)	M5Rules	evaluadosc parking - (6)
Storage	15,40(6,73)	IBK k=1	evaluadosc storage - (4)

Cuadro D.3: Comparativa evaluadosc

Longitud1

	Valor	Algoritmo	Fichero
Logistics	2,42(0,51)	IBK k=1,3,5	longitud1 logistic - (2)
Parking	17,02(2,49)	IBK k=1,3,5	longitud1 parking - (1)
Storage	9,21(8,54)	IBK k=1	longitud1 storage - (1)

Cuadro D.4: Comparativa longitud1

Longitud2

	Valor	Algoritmo	Fichero
Logistics	26,57(2,87)	IBK k=3	longitud2 logistic - (3)
Parking	20,80(3,21)	M5Rules	longitud2 parking - (5)
Storage	1,62(0,29)	IBK k=1,3,5	longitud2 storage - (6)

Cuadro D.5: Comparativa longitud2

Longitudc

	Valor	Algoritmo	Fichero
Logistics	3,77(0,38)	M5Rules	longitudc logistic - (5)
Parking	59,22(20,46)	M5Rules	longitudc parking - (5)
Storage	39,01(11,60)	IBK k=1	longitudc storage - (6)

Cuadro D.6: Comparativa longitudc