UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA: INGENIERÍA EN SISTEMAS

TEMA:

"ESTUDIO DE LAS TÉCNICAS DE PLANNING PARA LA BÚSQUEDA DE SOLUCIONES BASADAS EN HEURÍSTICAS"

Tesis previa a la obtención del Título de Ingeniero de Sistemas

AUTORES:

Ruth Natalia Ortiz Pacheco.
Darwin Stalin Ruilova Gonzalez.

DIRECTOR:

Ing. Vladimir Robles Bykbaev.

CUENCA - ECUADOR

2012

Breve reseña del autor e información de contacto

Darwin Stalin Ruilova Gonzalez

Estudiante de la Carrera de Ingeniería de Sistemas

Universidad Politécnica Salesiana

stalin_2003@hotmail.com

Ruth Natalia Ortiz Pacheco

Estudiante de la Carrera de Ingeniería de Sistemas

Universidad Politécnica Salesiana

nata.36@hotmail.com

Ing. Vladimir Robles B.

CERTIFICA

Haber dirigido y revisado prolijamente cada uno de los capítulos del informe de tesis, realizado por la Sra. Ruth Natalia Ortiz Pacheco y el Sr. Darwin Stalin Ruilova Gonzalez, y por cumplir los requisitos autorizo su presentación.

Cuenca, 06 de Septiembre del 2012

Ing. Vladimir Robles B.

Director de Tesis

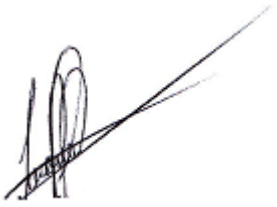
DECLARATORIA DE RESPONSABILIDAD

Los autores declaramos que los conceptos desarrollados, análisis realizados, y las conclusiones del presente trabajo, son de nuestra exclusiva responsabilidad y autorizamos el uso a la Universidad Politécnica Salesiana el uso de la misma con fines académicos.

A través de la presente declaración cedemos los derechos de propiedad intelectual correspondiente a este trabajo a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Cuenca, 06 de Septiembre del 2012

Ruth Natalia Ortiz Pacheco



Darwin Stalin Ruilova Gonzalez

DEDICATORIA

Este trabajo de tesis está dedicado con todo mi cariño a mi padre Vicente Ortiz por brindarme su apoyo, a mi madre Anita Pacheco por todo su esfuerzo y sacrificio constante, digno de admirar para que yo pueda alcanzar tan esperado triunfo.

A mi esposo Stalin Ruilova, gracias por tu amor tu paciencia por darme siempre fuerzas para continuar, que orgullo tan grande de haber luchado juntos y finalmente alcanzado nuestro sueño.

A mi hija Paula Valentina mi pequeña doncella, porque tu presencia en mi es el mejor regalo que me dio la vida, tan chiquita, tan hermosa, tan perfecta, todo lo que hoy soy es por ti hija mía, porque con cada sonrisa, cada gesto, cada palabra, cada llanto iluminas mi mundo, que bendición tan grande tenerte junto a mí y que me digas mama.

A mis queridos abuelitos José Leoncio y Luz María, pilares fundamentales en mi vida, gracias por su amor, consejos y enseñanzas, han fomentado en mi todo lo que hoy soy, gracias por regalarme momentos únicos y experiencias hermosas que siempre llevo en mi corazón, Uds. son el complemento para mi felicidad, que nuestro Señor les preste la vida para que estén junto a mí por muchos años más. Mil gracias amados papitos.

A mi hermana Gemita por estar junto a mí y formar parte de mi mundo como mi hermana, amiga y cómplice.

A mis tíos Edgar, Alfredo y Eulalia por brindarme una palabra de aliento y estar presentes en cada momento de mi vida.

A mi pequeña Nicol por estar siempre pendiente de mí, por cada llamada por cada gesto que simplemente representa un cariño tan puro y sincero.

Natalia Ortiz

DEDICATORIA

La presente tesis está dedicada a mis queridos padres Darwin Ruilova y Yolanda Gonzalez que me han brindado todo su apoyo y amor a lo largo de mi vida, convirtiéndose en la base fundamental para lograr este objetivo. A mi querido Dios que ha estado siempre presente en mí, recordándome que con humildad, esfuerzo y apoyo suyo podre lograr muchas metas importantes en mi vida.

A mi amada esposa Natalia Ortiz, compañera de tesis y de mi vida, con quien he pasado momentos muy especiales, gracias por estar a mi lado y darme el regalo más hermoso, Valentina.

A mi querida hija Valentina que es el motor que impulsa mi vida, que me da aliento para seguir adelante, la fuerza para alcanzar metas, el verte crecer y que estés junto a mí ha sido el regalo más hermoso que me ha podido dar Dios, gracias a ti mi hija amada he podido coger fuerzas y salir adelante frente a todos los obstáculos y contratiempos, para ti mi "Valentinita", mi "Gordita", va dedicado este trabajo.

A mis hermanos Andrés y Lizbeth que siempre han estado a mi lado ayudándome y siendo mis mejores amigos, los quiero mucho hermanos, gracias por su apoyo incondicional.

Para mis abuelitas Marina Pereira y Teresa Jaramillo, que siempre me han dado su amor y comprensión, muchas gracias son cosas que nunca se borraran de mi mente y corazón. Además a todos mis primos, primas, tíos y tías muchas gracias por estar a mi lado.

Finalmente para todos mis amigos que han estado a mi lado, con los que pasamos muchas vivencias, que reímos, que cantamos, que lloramos, a todos ellos muchas gracias por estar ahí.

Stalin Ruilova

AGRADECIMIENTOS

Agradezco primeramente a Dios por darme salud y vida y la dicha de estar junto a todos mis seres querido, a mis padres Vicente Ortiz y Anita Pacheco por apoyarme en todo momento, su paciencia y esfuerzo están reflejados en este trabajo.

A mi esposo y compañero de tesis, Stalin Ruilova, muchas gracias por estar a mi lado y darme tú apoyo, aliento y paciencia.

A mis queridos Abuelitos Leoncio y Luz María, mis segundos padres, muchas gracias por todo el apoyo, consejos y amor brindados, les agradezco de todo corazón

Gracias a nuestro tutor Vladimir Robles, por su gran paciencia, preocupación, y entendimiento, por su gran e inmenso apoyo, por el tiempo que nos ha dedicado, pese a estar muy atareado, se dio el tiempo y la dedicación necesaria para que esta tesis salga adelante, le agradezco mucho y sepa que pueda contar conmigo para lo que sea, más que un docente o tutor de tesis usted se a convertido en un gran amigo.

Finalmente gracias a mis amigos y amigas que estuvieron a mi lado dándome fuerza y aliento para no retroceder, sería injusto nombrar a unos cuantos, ya que son muchos los que me ayudaron, gracias a ustedes adquirí muchos conocimientos que me sirvieron a lo largo de la carrea y vida .

Natalia Ortiz

AGRADECIMIENTOS

Agradezco en primer lugar a Dios por brindarme la vida y salud para poder disfrutar de este logro, a mis queridos padres Darwin Ruilova Y Yolanda Gonzalez, por estar a mi lado siempre pese a muchas dificultades han estado siempre a mi lado, aconsejándome y guiándome, esas son cosas que siempre guardare en mi mente y corazón, siempre encaminándome por el rumbo correcto, gracias a ustedes he podido culminar este trabajo.

A mí amada esposa Natalia, compañera de vida y de tesis, muchas gracias por estar junto a mi ayudándome en todo momento, has sido un pilar fundamental en la conclusión de esta tesis, teniéndome paciencia, ayudándome a sobreponerme y a sobrellevar problemas. A mi pequeñita hija Valentinita, que con toda su ternura me brindo la fuerza necesaria para continuar y no desfallecer.

A mis queridos hermanos Andrés y Nena, gracias hermanos siempre han estado a mi lado, poniendo las manos al fuego por mí, apoyándome y siempre dándome ánimos.

A nuestro tutor y amigo el Ing. Vladimir Robles muchas gracias por todo el tiempo invertido en nosotros, por toda su paciencia, estoy orgulloso de que usted haya sido nuestro tutor, y mentor para que este proyecto salga adelante, siempre preocupado, le agradezco de todo corazón.

Finalmente a todos mis amigos y compañeros de la universidad, gracias por su ayuda y apoyo, gracias a ustedes he podido adquirir conocimientos y aplicarlos, no solo en el lapso de la universidad, sino en todo momento de mi vida.

Stalin Ruilova

INDICE DE CONTENIDO

INDICE DE IMAGENES

ÍNDICE DE ECUACIONES

INDICE DE TABLAS

CAPITULO I

1.1 Introducción

El tema que se plantea en esta tesis es el estudio de las técnicas de *planning* para la búsqueda de soluciones basadas en heurísticas, el cual se aplica en el mundo real, como la robótica, fabricación mediante ensamblado de componentes, misiones espaciales [1], agentes autónomos, planificación de procesos etc. [1]. La planificación es un fundamento clave en la Inteligencia Artificial ya que se basa en el razonamiento para posibles eventos futuros en el proceso de aprendizaje e investigación.

El punto de partida para la planificación surgió hace tres décadas, al crear un sistema con la finalidad de controlar un robot llamado *Shakey* [2] que merodeaba por los espacios del SRI (*Standford Research Institute*), a raíz de ello se diseñó el primer lenguaje de representación *STRIPS (Stanford Research Institute Problem Solver*), considerado desde entonces como base para la evolución de los demás lenguajes ahora conocidos, entre ellos NOAH, SNLP, O-Plan, UCPOP, etc. [2].

La planificación consiste en elaborar un plan[3] para la resolución de cualquier problema ya que en él se determina cada paso que busca llegar de manera consecutiva a un estado final [2]. Por lo tanto, el objetivo de la planificación está en explorar el espacio de estados de la mejor manera posible y poder reducir el costo realizado por la búsqueda.

Esta técnica tiene como ventaja, darnos una idea clara de los fines que se pretende alcanzar, estimando el costo y tiempo de duración que conlleva resolver un problema.

---

[1] Mars Exploration Rovers, la planificación de las tareas a realizar durante un día marciano se realiza automáticamente por un programa a partir de los objetivos de exploración que fija el personal de misión en la Tierra.

[2] Es un robot que se puede mover entre varias habitaciones, empujar objetos, trepar a objetos rígidos, encender y apagar las luces. Fuente: http://search.conduit.com/Results.aspx?q=Planificacion+Maite+Urretavizcaya+Isabel+Fern%C3%A1ndez&Suggest=&stype=Homepage&SelfSearch=1&SearchType=SearchWeb&SearchSource=10&ctid=CT3148764&octid=CT3148764

[3] Plan.- Es una secuencia de acciones que, cuando se ejecuta desde el estado inicial produce un estado que satisface el estado objetivo. Fuente: Wikipedia

Si bien es cierto que por los años 90si se elaboraban planes con una secuencia de acciones, pero no se consideraba un orden parcial de los pasos a seguir, por lo que no se podía generar más de un cierto número de acciones, a raíz de ello surge la planificación basada en grafos independientes de dominio *Graphphan* por el año 1995 [2].

Para el año 2000 surgen mejoras muy importantes que permiten resolver problemas complejos como *Metric-FF (Fast Forward)[42],* LPG*(Local Search for Planning Graphs)*oSGPlan *(Subgoal Partitioning and Resolution in Planning)* que ofrecen soluciones sub-optimas independientes del dominio [2]. Mediante estos planificadores se elaborar soluciones basadas en la exploración de nodos en problemas NP-Completos, siguiendo una secuencia de acciones a través de un proceso de búsqueda capaz de alcanzar un objetivo planteado.

Es importante mencionar que los fundamentos de cómo planificar son los siguientes [3]:

- Establecer metas y los medios.
- Determinar jerarquías y establecer prioridades.
- Descomponer problemas en sub-problemas.
- Basado en la experiencia.

La representación de los problemas de planificación debe admitir que los algoritmos que se utilizan experimenten todas las posibles acciones a desarrollar en cada dominio del problema, ya que de ellos depende que el proceso de planificación se pueda implementar en la práctica, de manera que sean eficientes y capaces de describir de manera lógica un amplio rango de problemas, utilizando funciones heurísticas[4], que a su vez están guiadas por una función de evaluación[5] definida como heurística, siendo estas de gran ayuda en los últimos años.

Asimismo, en los últimos años se presentó una técnica inductiva, la cual permite acceder a modificar las desviaciones del coste real del valor heurístico, este tema se profundiza en el capítulo II.

[4] Función Heurística.- disponen de alguna información sobre la proximidad de cada estado a un estado objetivo, lo que permite explorar en primer lugar los caminos más prometedores. Fuente: http://www.nebrija.es/~cmalagon/ia/transparencias/busqueda_heuristica.pdf

[5] Función de Evaluación.- Dado un nodo *n* estima la distancia desde ese nodo *n* a un nodo objetivo. Fuente: http://www.aic.uniovi.es/ssii/SSII-T3-BusquedaII.pdf

La organización de esta tesis consta de cinco capítulos, conclusiones y anexos, cuya estructura y contenido se presenta de la siguiente forma:

En el capítulo I se hace un breve recorrido con las ideas principales del tema. De igual forma, se desarrolla la justificación y el motivo de la investigación en el que se definen porque es necesario llevar a cabo este trabajo, también se elabora un análisis de la problemática planteada y de este análisis se determinan los objetivos de la tesis.

El capítulo II se centra en la tarea que tienen las funciones heurísticas para guiar correctamente la búsqueda del planificador, se realiza un análisis de la planificación clásica que se desarrolla sobre un entorno de aplicación, además el estudio de los algoritmos de planificación así como una descripción detallada de los mismos junto con las técnicas a utilizar para hacer más eficiente la búsqueda.

El capítulo III y IV plantean el aprendizaje aplicado a la planificación, también se enfatiza en los conceptos más relevantes de los puntos de control, retroceso y posteriormente las funciones de evaluación. Se definen también detalles de la planificación basada en casos y en el espacio de estados, igualmente se expone un análisis de la planificación numérica, jerárquica y por incertidumbre.

El problema práctico, que se desarrolla en el capítulo V, busca aplicar algoritmos basados en heurísticas que encuentren soluciones eficaces con diversas técnicas para la planificación, asimismo, se expondrá un análisis y discusión del tema tratado.

Finalmente, se presentarán las conclusiones de la investigación llevada a cabo y los anexos.

1.2 Justificación

En la actualidad es fundamental encontrar soluciones óptimas e inteligentes a problemas, donde se apliquen eficientes acciones para pasar de un estado inicial a un estado objetivo, tomando en cuenta posibles escenarios ya planteados. Ya que hoy en día existen muchos problemas que no se pueden resolver con la programación tradicional o convencional, la planificación se ha ido abriendo campo en el área de la informática y se ha convertido en una

técnica muy útil e interesante, ya que resuelve problemas de alta complejidad, con eficacia y eficiencia, además de reducción de tiempo en su implementación comparada con otras técnicas.

La metodología de planificación (*planning*), ayuda a encontrar soluciones entre una serie de sucesos y opciones, tomando como parte fundamental la elección de un algoritmo que satisfaga las necesidades del problema planteado. Además se pretende aplicar técnicas de planificación para mejorar una solución existente, buscando el razonamiento apoyado en reglas que necesitan heurísticas para lidiar con información incompleta y/o compleja[6].

En la actualidad en el Ecuador no existen soluciones que se basen en esta metodología, por esta razón es que hemos optado por la misma, con lo se pretende demostrar que con la ejecución de el o los algoritmos correctos, se lograra optimizar la utilización de recursos limitados, solucionando el problema, disminuyendo factores fundamentales los cuales inciden de gran manera sobre el tema en discusión.

1.3 Definición de Objetivos

1.3.1 Objetivo General

Conocer y estudiar las técnicas de *planning* a fin de solucionar un problema de aplicación real.

1.3.2 Objetivos Específicos

- Investigar cada una de las diferentes técnicas que existen de *planning.*
- Comparar las técnicas en cuanto a ventajas y desventajas de cada una ellas.
- Conocer las estructuras de las técnicas que permitan planificar el diseño del sistema.
- Investigar qué tipo de software existe para poder realizar la implementación.

---

6 http://translate.google.com.ec/translate?hl=es&sl=en&u=http://aaai.org/AITopics/Planning&ei=vqWDT6PaLpLrtgfdiqXaBw&sa=X&oi=translate&ct=result&resnum=4&ved=0CEIQ7gEwAw&prev=/search%3Fq%3Dplanning%2Band%2Bscheduling%2BIA%26hl%3Des%26biw%3D1366%26bih%3D645%26prmd%3Dimvns

- Plantear un problema de aplicación real y resolverlo a través de técnicas de *planning* y medir cuan óptimas son las soluciones obtenidas.

1.4 Descripción de la problemática planteada

En este apartado se analizará los elementos que intervienen en la descripción de un problema para encontrar una solución a través de la construcción adecuada.

Para ello es necesario especificar los siguientes elementos [5].

- El objetivo que se busca conseguir.
- El estado inicial del entorno del problema.
- Conjunto de acciones a cumplir.

Para una mayor explicación se representa un problema mediante la siguiente tupla [4]:

$$P = (I, G, O)$$

Ecuación1: Define un problema de planificación

En donde:

P → Detalla los hechos relevantes del problema

O → Representa el conjunto de acciones

I → Es el estado inicial del problema

G → Es el estado final del problema

En la ilustración 1 se puede observar como estos elementos intervienen al momento de definir un problema.

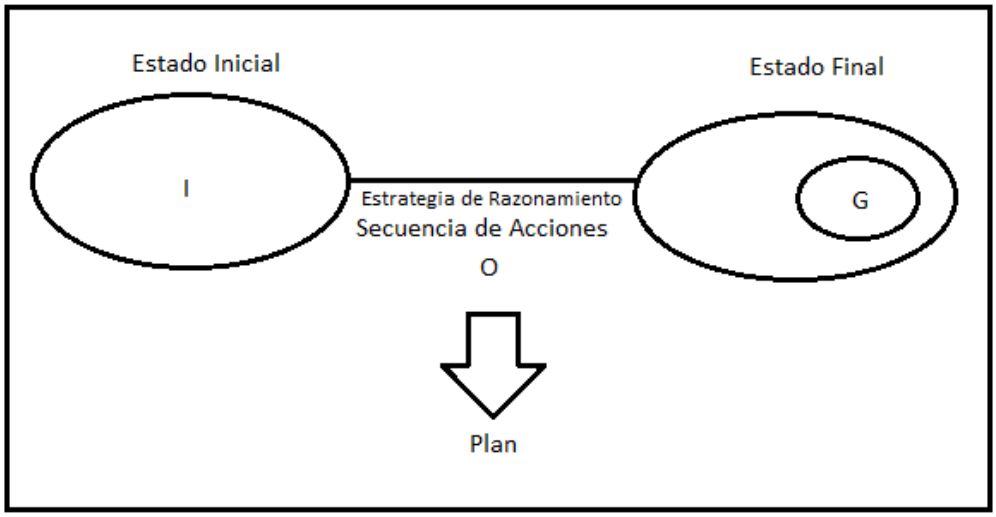


Ilustración 1. Esquema general para la definición de un problema de planificación [2].

Partiendo de estos puntos, es necesario construir algoritmos y proporcionar técnicas basadas en el razonamiento inteligente que permitan a un agente estudiar esa secuencia de acciones para conseguir una meta.

Como ya se mencionó en la introducción, cada problema de planificación se representa mediante un lenguaje que describa los estados, en este caso el lenguaje *STRIPS* es considerado por definir los problemas de una manera práctica.

A continuación se da una breve explicación de la sintaxis en este lenguaje:

- El estado inicial es una descripción detallada para partir a la solución.
- El estado final, es el objetivo del problema.
- Una acción toma como argumento un estado inicial para poder determinar cuál será el siguiente estado y dependiendo del tipo de problema se determinan el número de condiciones.
- Una acción consta a su vez de:
    - Nombre y lista de argumento.-
    - Precondiciones
    - Efectos

Finalmente, el plan al que se pretende llegar es la solución a un problema, siendo una secuencia de acciones que se establece desde el inicio hasta el estado objetivo [4].

Ilustraremos mejor estos conceptos con el siguiente ejemplo, suponga que se tiene la rueda de un auto deshinchada, por lo que es necesario que se cambie la rueda dañada, entonces, se parte de un estado inicial que en este caso sería la rueda deshinchada sobre el eje del auto y la rueda nueva que se encuentra ubicada el maletero, por lo tanto el objetivo es colocar la rueda nueva sobre el eje del auto de manera apropiada y precisa [46].

Consta de las siguientes acciones. [11]:

- Sacar la rueda de nueva que sirve como repuesto del maletero.

  Acción (Quitar, (Repuesto, maletero))
  Precond: En (Repuesto, maletero)
  Efecto: ¬En (Repuesto, maletero) ^ En (Repuesto, suelo))

- Quitar la rueda deshinchada del eje

  Acción (Quitar, (Deshinchada, Eje)),
  Precond: En (Deshinchada, Eje)
  Efecto: ¬En (Deshinchada, Eje) ^ En (Deshinchada, suelo))

- Colocar la rueda de repuesto en el eje

  Acción (Colocar, (Repuesto, eje)),
  Precond: En (Repuesto, suelo) ^ ¬En (Deshinchada, eje)
  Efecto: ¬En (Repuesto, suelo) ^ En (Repuesto, eje)

- Dejar el coche por la noche solo y sin vigilancia

  Acción (DejardeNoche,
  Precond:
  Efecto: ¬En (repuesto, suelo) ^ ¬En (repuesto, eje) ^ ¬En (repuesto, maletero) ^ ¬En (deshinchada, suelo) ^ ¬En (deshinchada, eje)

Iniciar (En (Deshinchada, Eje) ^ En(repuesto, maletero))
Objetivo (En (repuesto, eje))
Acción (Quitar, (Repuesto, maletero))
Precond: En (Repuesto, maletero)
Efecto: ¬En (Repuesto, maletero) ^ En (Repuesto, suelo))
Acción (Quitar, (Deshinchada, Eje)),
Precond: En (Deshinchada, Eje)
Efecto: ¬En (Deshinchada, Eje) ^ En (Deshinchada, suelo))
Acción (Colocar, (Repuesto, eje)),
Precond: En (Repuesto, suelo) ^ ¬En (Deshinchada, eje)
Efecto: ¬En (Repuesto, suelo) ^ En (Repuesto, eje)
Acción (DejardeNoche,
Precond:
Efecto: ¬En (repuesto, suelo) ^ ¬En (repuesto, eje) ^ ¬En (repuesto, maletero) ^ ¬En (deshinchada, suelo) ^ ¬En (deshinchada, eje)

Ilustración 2. Ejemplo de la rueda de cambio [5].

Ahora bien:

- Una acción es válida si satisface las precondiciones de un estado.
- A su vez, cuando la acción es válida se produce un efecto, dando como resultado el cambio de estado.
- Si una Acción no es válida se entiende entonces que es falsa.
- Asimismo, un efecto representa el cambio de estado que se da al ser ejecutada una acción.

A diferencia del lenguaje *STRIPS (Stanford Research Institute Problem Solver)* el lenguaje de PDDL (Planificación de dominio del lenguaje de definición), contiene extensiones como ADL (Lenguaje de descripción de acciones) [46],con una representación de los problemas mucho más completa, la misma que incluye predicados negados, cuantificadores y efectos condicionales.

Un problema se describe en lenguaje PDDL por [6]:

- **Por el Dominio.-** Indica el espacio de estados por medio de tipos, constantes y predicados, así como la descripción de sus acciones, las mismas que incluyen las precondiciones, complementos del problema y borrados de ser necesario.
- **Por el Problema.-** Especifica el estado inicial, el objetivo alcanzar y los objetos del problema.

Por tanto, la planificación busca construir algoritmos de control que permitan a un agente resumir la secuencia de acciones que lleven a lograr objetivos deseados por el planificador. Asimismo, proporciona una base para la recuperación de errores así como la disminución del tiempo de búsqueda a fin de que esta no sea muy extensa .De igual manera, al presentarse dificultades con los objetivos planteados, ayudan a encontrar una solución y de esta manera se demuestra que tan eficiente e importante es la descripción de una problemática para la aplicación de *planning.*

1.4.1 Lenguajes de Problemas de Planificación

Para la solución de un problema se debe aplicar un lenguaje según las especificaciones de cada necesidad, cada lenguaje cumple con las siguientes fases a seguir [12]:

- **Representación de objetivos.-** Cada objetivo que se describe, esta. representado en un estado.
- **Representación de estados.-** Cada estado está representado por un conjunto de literales positivos.
- **Representación de acciones**.- Las precondiciones se tienen que cumplir para que se produzca una acción y se pueda ejecutar el plan produciendo así un efecto por ejemplo:
    - Precondiciones.- Cada precondición que se plantee, tiene que ser verdadera para que se ejecute una acción
    - Efecto.- Es el resultado de ejecutar la acción y por lo tanto se presenta un cambio en el estado.

CAPITULO II

2.1 Funciones Heurísticas

Las funciones heurísticas son de gran importancia en la planificación, y aportan gran conocimiento para la Inteligencia Artificial, por lo que es realmente interesante su estudio en este capítulo, ya que de la elección de una buena heurística hará que la implementación de un problema sea exitosa.

Previo a la descripción de las funciones heurísticas, es básico definir primero una heurística, tal como se indica en [7]:

Las heurísticas nos indican cómo se debe elegir una búsqueda para que esta resulte fiable en un espacio compuesto por estados[7] con problemas NP-Complejos.

Básicamente, las heurísticas se pueden presentar de dos formas:

- Se pueden dar varias interpretaciones de un problema, lo que hace que no se tenga una solución exacta.

  Ejemplo: Diagnóstico médico

- El costo computacional puede ser una gran desventaja para encontrar una solución exacta.

  Ejemplo: Juego de Ajedrez [8].

En tanto, las funciones heurísticas saben aprovechar al máximo toda la información que poseen para evitar pérdidas de tiempo al realizar búsquedas innecesarias, es decir, se basan en el conocimiento[8] que tienen del dominio, utilizando el mismo para encontrar una solución.

Ahora bien, cuando un problema se presenta como un árbol de búsqueda, la tarea de las funciones heurísticas se centra en guiar la búsqueda analizando la distancia entre los caminos más significativos en un espacio de estados y así poder reducir el tamaño de un árbol al ir podando los nodos poco prometedores, también se estima el costo de la función heurística para lograr una solución mucho más directa en el proceso [32].

[7] Estado.- Una posible configuración de un problema.

[8] Todo el conocimiento específico que se va a usar sobre el problema está codificado en la función heurística. Fuente: http://www.cs.us.es/cursos/ia1/temas/tema-04.pdf

A su vez se puede maximizar o minimizar su valor de acuerdo al nodo que se necesita encontrar, por ejemplo, si se busca el estado más cercano, convendría entonces minimizar su valor, caso contrario dependiendo de la necesidad se puede maximizar el valor [10].

Una Función heurística se representa de la siguiente forma [49]:

- $h(n) =$ Estima el costo alcanzar desde el nodo n hasta el objetivo
- Si n es el objetivo, entonces $h(n) = 0$

Asimismo, presenta la ventaja poder utilizar esta misma información en otros sistemas, por lo que la inteligencia artificial ha hecho uso de estas funciones para resolver ejemplos como el *8-Puzel*, el problema del viajante [21].

Como se puede ver en la ilustración 3 se muestran algunos ejemplos en los cuales se ha aplicado funciones heurísticas:

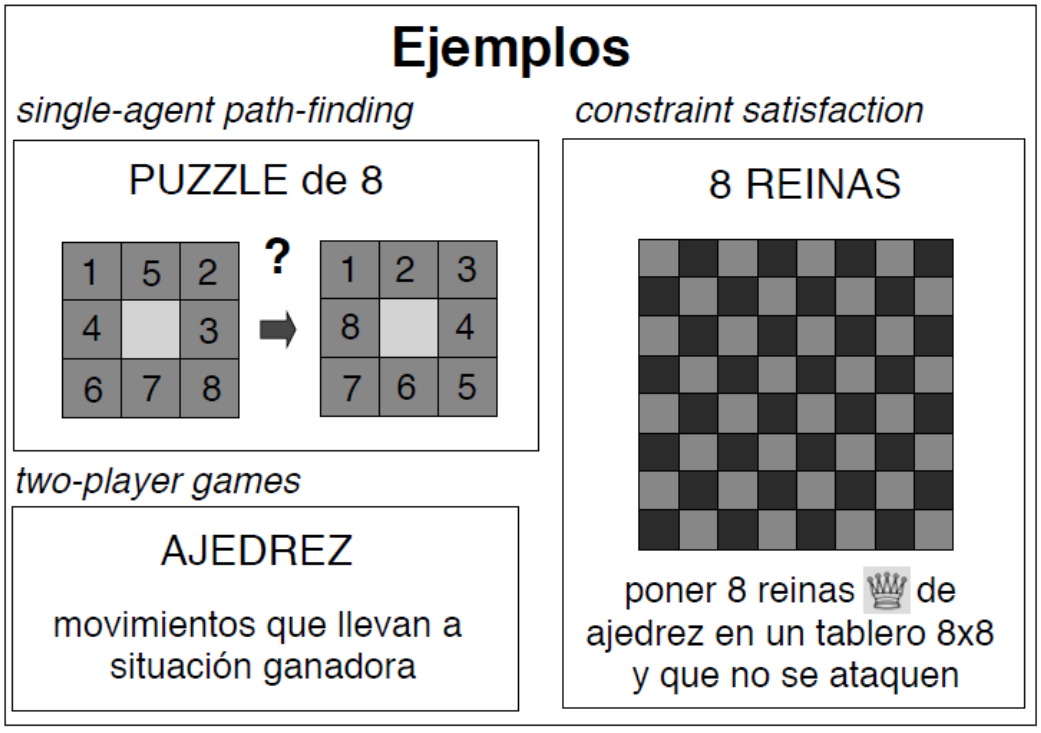


Ilustración 3. Ejemplos que han aplicado funciones heurísticas [9].

Como un apartado importante se establece fundamental que un planificador pueda definir las heurísticas adecuadas y dar un valor a cada variable del

problema ya que requiere un gran estudio al tratarse de problemas NP-complejo.

Es necesario tener en cuenta los siguientes puntos para encontrar una buena función [22]:

- Se debe considerar el costo que tiene la búsqueda.
- La evaluación heurística debería de conseguir el propósito planteado.
- Se puede utilizar un problema relajado[9] para determinar la distancia hasta el objetivo, ya que en general es mucho más fácil de resolver debido a que puede obviar las restricciones y efectos negativos que se imponen a los operadores. Además de esto, tiene como ventaja encontrar una solución óptima estimando su costo, de esta manera puede definir una buena heurística para aplicarla en el problema principal.
- Resulta ventajoso funciones que se puedan maximizar.

2.1.1 Funciones Genéricas.

"S*on adecuadas para una amplia variedad de dominios. Por ejemplo, la heurística del vecino más próximo (nearest neighbor) se aplica a muchos problemas combinatorios***"** [45].

2.1.2 Funciones Reutilizables

Son muy útiles ya que aprovechan la información que tienen de un plan elaborado para utilizarla en otros procesos, lo que es ventajoso en cuanto a tiempo y costo en problemas complejos**.**

2.1.3 Función de Evaluación

La funcione evaluación tienen como finalidad, ayudar en la decisión de expandir o no un nodo, ya que los nodos se ordenan de acuerdo a los valores que se obtienen de aplicar esta función, por lo que, se debe escoger el nodo con menor costo para que su calidad sea mucho mejor al llegar a la meta, es

[9]Problema Relajado.- Consistía en el mismo estado inicial y las mismas metas, pero la aplicación de las acciones solo añadían nuevos literales al estado.

importante que el planificador tenga un claro conocimiento sobre el problema [17][50].

La calidad de un nodo se puede determinar por lo siguiente [51]:

- Dificultad de buscar la solución al sub-problema que representa cada nodo.
- La calidad que presentan las posibles soluciones codificadas en un nodo. Cantidad y calidad de información que se pueda generar al expandir un nodo.

Como se puede ver en la ilustración 4 el árbol de búsqueda tiene un nodo A que a su vez se expande los nodos B y C y estos en varios nodos de acuerdo a su conveniencia de acuerdo a la decisión de la función de evaluación.

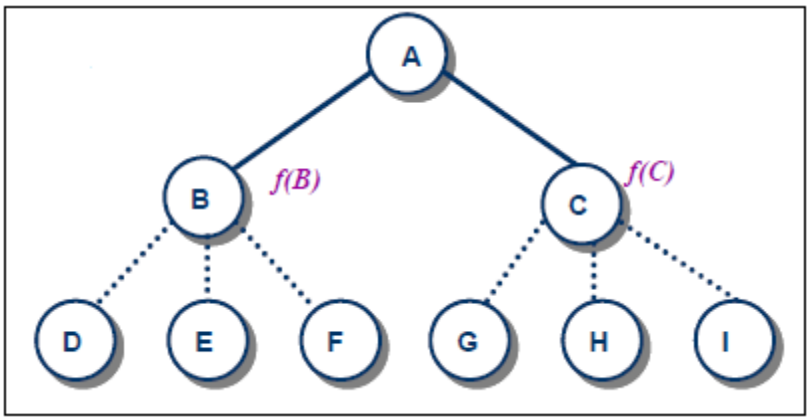


Ilustración 4. Árbol de búsqueda que expande sus nodos a su conveniencia, según la función de evaluación [7].

La función de evaluación es la siguiente:

$$f(n) = g(n) + h(n)$$ [47].

Ecuación 2: Función de evaluación

Donde [47]:

n → Es el estado

$g(n)$ → Estima la longitud de un camino hasta alcanzar la meta

$h(n) \rightarrow$ Estima el valor heurístico

El valor a elegir para la función de evaluación debe ser el más bajo para que se cumpla con $h(objetivo) = 0$ [7].
Algunas características de la función de evaluación son [17]:

- La función puede crecer o decrecer según como se vaya aproximando a la meta.
- Según las necesidades, la función determina las estrategias, realizando movimientos de acuerdo la acción propuesta por la función de evaluación.
- El valor de su función dura mientras se encuentra el camino óptimo.

2.2. Planificación clásica

En este apartado analizaremos los principios de la planificación clásica que ha sido la base para muchas investigaciones. Esta planificación tuvo sus indicios por el año de 1963 con el planificador GPS (*General problem solver*)[10] y luego por los años 70 surgió el lenguaje de planificación *STRIPS* [11], con sus primeras aplicaciones con respecto a la búsqueda(que en sus inicios fue vista como una exploración del espacio de estados en un determinado entorno).
La planificación clásica se estructura por los siguientes componentes [53], tal como se puede observar en la ilustración 5:

- La descripción de un estado inicial
- La descripción del conjunto de objetivos
- El conjunto de posibles acciones.

[10] Un razonador que trataba de resolver cualquier problema de manera general, mediante una técnica de búsqueda basada en el análisis de medios y fines. Fuente: http://hera.ugr.es/tesisugr/16795143.pdf

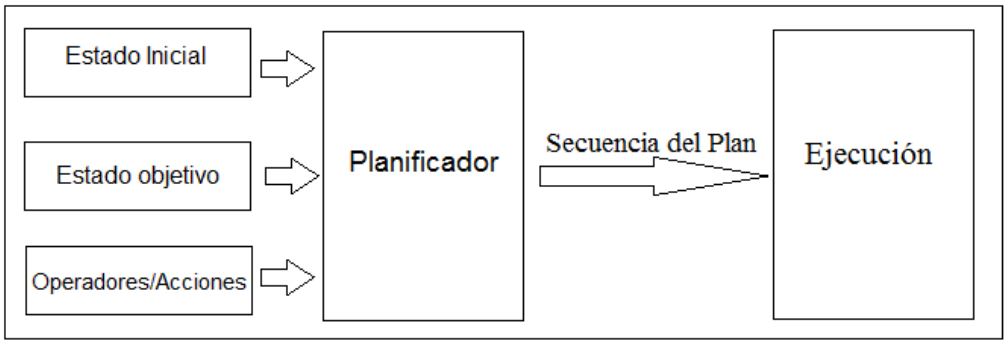


Ilustración 5. Estructura de un sistema de planificación clásica [53].

Los planificadores prefieren trabajar con este modelo de planificación clásica, ya que les proporciona información y se considera con entornos únicos, descritos a continuación [12]:

- **Observables.-** El planificador conoce todos los hechos principales en su entorno y el efecto de sus acciones en el mismo.
- **Deterministas.-** En este modelo sus acciones están bien especificadas para predecir y predefinir sus efectos.
- **Finitos.-** Modelo en donde existen acciones y estados definidos
- **Estáticos.-** Este modelo cambiará solo cuando el agente planificador actué sobre el mismo, ya que se debe tener en cuenta que antes de comenzar la planificación se conoce los objetivos y estos no cambiaran en el transcurso del proceso de planificación.
- **Discretos.-** En cuanto a l tiempo, acciones, objetos y efectos.

Esta planificación utiliza elementos importantes como acciones y estados, ya que cada problema se representa por un modelo formado por estados, cada uno de estos estados se pueden ver como una instancia del mundo en un tiempo determinado y a su vez cada modelo de estados se caracteriza por lo siguiente [53]:

- Un espacio de estados
- Un estado inicial
- Un conjunto de estados meta
- Un conjunto de acciones que sean aplicables a cada estado
- Una función de transición

- Costes de ejecución para las acciones

Cabe indicar que este modelo de planificación presenta una ventaja muy importante, que es poder transformar problemas planteados en otros modelos a este, con el objetivo de llegar a la meta por medio de una serie de acciones desde su estado inicial [6].

En tanto, en estos últimos años han surgido nuevas técnicas de planificación, tales como [13]:

- **Planificación Condicional.-** El planificador no necesita tener toda la información del problema, debido a que examina el modelo actual para en base a este determinar la acción que se va a tomar.
- **Planificación Continua.-** Primero realiza un estudio sobre el estado en el que se encuentra el problema para reevaluar los objetivos ya planteados son el propósito de satisfacer las cada uno de los objetivos hasta alcanzar la meta.
- **Planificación Basada en el Objetivo.-** Básicamente esta planificación ayuda a establecer las necesidades que tiene el usuario para poder establecer un plan de un problema.

Actualmente se utilizan planificadores[11] que buscan encontrar soluciones sub-optimas, entre ellos tenemos a *Metric-FF* un sistema independiente de dominio[12] o *SGPlan* que calcula soluciones rápidas. El lenguaje que se utiliza para su representación se basa en el lenguaje *STRIPS*, siendo este uno de los primeros planificadores que planteó algoritmos simples y eficientes para evitar dificultades de problemas reales.

De igual manera, maneja lógica de predicados con una sintaxis poco compleja para la elaboración de problemas de dominios de planificación y debido a su capacidad expresiva aprovechan su estructura evitando complejidad en los problemas, ya que los únicos cambios que se realizan son los efectos que se dan cuando se cumple una acción[4][6].

[11] Tratan de encontrar una secuencia de acciones (plan) que haga verdadera la meta. Fuente: http://www.smartcomputing.com.ar/representacion-mediante-strips.aspx

[12] El sistema es una extensión de FF para el planificador (ADL combinado con) las variables de estado numéricos, más precisamente a PDDL. Fuente: http://www.loria.fr/~hoffmanj/metric-ff.html

Un problema de *STRIPS* trabaja con literales de primer orden y se define mediante la siguiente tupla [53]:

$$\boldsymbol{P} = (\boldsymbol{A}, \boldsymbol{O}, \boldsymbol{I}, \boldsymbol{G})$$

Ecuación 3: Define un problema *STRIPS*

En donde:

A → Conjunto de variables de estados

O → Conjunto de operadores

I → Estado inicial

G → Estado final

STRIPS es un lenguaje que busca sobre un espacio de modelos del mundo, localizar un espacio donde se encuentre la solución, a través de la composición de un conjunto de operadores capaces de transformar el modelo inicial en un modelo capaz de satisfacer el objetivo del problema [54].

Un operador o acción está compuesto de la siguiente manera [15]:

- **Especificación de la Acción.-** Determina el nombre que tiene la acción y los argumentos que van a intervenir.
- **Precondiciones.-** Es necesario que las condiciones sean verdaderas y se cumplan para que una acción pueda sea ejecutada y válida, caso contrario es falsa no tiene validez.
- **Efectos.-** Un efecto se produce cuando se ejecuta una acción, es importante indicar que un hecho se puede agregar (*Addlist*) o se puede quitar (*Dellist*) según la consecuencia que tenga cada acción.

A continuación se presenta un ejemplo básico para llenar un tanque con agua y que su flujo se mantenga constante por medio de una válvula tal como se puede ver en la ilustración 6 [54]:

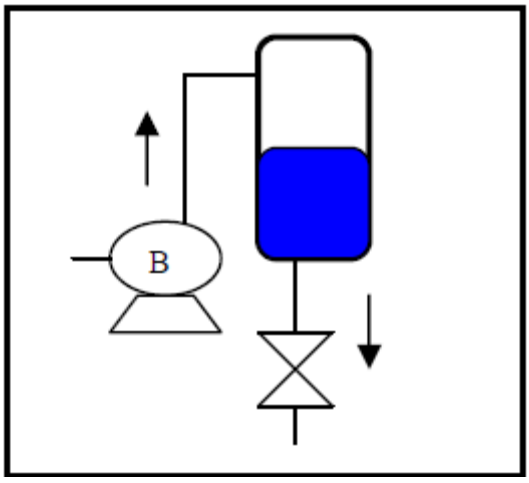


Ilustración 6. Problema que representa el abastecimiento de un tanque con agua con flujo constante o continuo [54].

En donde se puede definir como [54]:

Estado inicial:(bomba-apagada, tanque-vacío, válvula-cerrada)

Estado objetivo:(flujo constante)

Acciones:(llenar-tanque) ^(vaciar-tanque)

Precondiciones:(tanque-lleno) ^ (bomba-encendida)

Efectos: (*add-list*) (tanque-lleno), (*delete list*) (tanque-no lleno)

Cabe indicar que este lenguaje también se basa en la utilización de una pila para resolver problemas concretos, como es el caso del mundo de los bloques, de esta forma sus objetivos puede interactuar y a su vez cumplirse, por medio del siguiente proceso de pasos [14]:

- **Emparejamiento.-** Es necesario que el objetivo esté emparejado con el estado actual de la pila.
- **Descomponer.-** Agregar los literales componentes en la parte superior siempre que su objetivo en la parte superior sea compuesto.
- **Resolver.-** Se debe emparejar los objetivos para que los objetivos se cumplan.

- **Aplicación.-** La pila finalmente se suprime cuando se consigue un operador en su parte superior, lo que provoca que el estado actual se vea modificado y el nuevo estado es registrado en el plan para que el proceso pueda repetirse hasta que la pila se quede vacía.

En la ilustración 7 se muestra un ejemplo básico de cómo funciona el mundo de los bloques con planificación clásica.

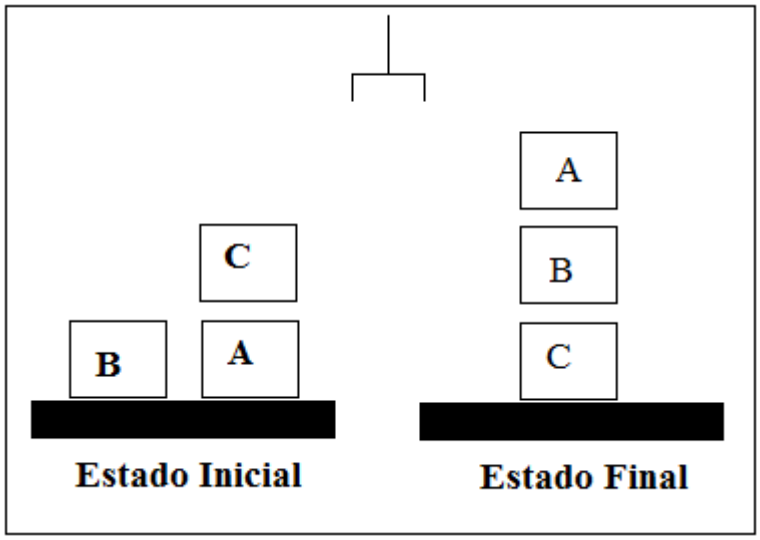


Ilustración 7. El problema de planificación (Anomalía de Sussman) en el mundo de los bloques [55].

Como se puede ver en el problema está formado por los siguientes elementos [1]:

- Un conjunto de bloques colocados sobre una superficie plana.
- Un brazo robótico, que realiza los movimientos con un solo bloque a la vez en un determinado tiempo.
- Todos los bloques son del mismo tamaño y forma.
- Cada bloque se distingue por su nombre.
- La superficie tiene una extensión ilimitada.
- Cada bloque se puede colocar sobre la superficie, encima de un bloque, o puede estar sujeto del brazo robótico.

Ahora bien cuando el brazo realiza un movimiento en un bloque se produce un cambio en el estado, pasando de un estado 1 a un estado 2, por lo que no

se puede tomar un nuevo bloque si no ha soltado primero el anterior. El proceso puede llegar hacer varios movimientos hasta alcanzar el objetivo planteado.

A lo largo del tiempo se han estudiado nuevas extensiones de este lenguaje, con mejoras muy significativas entre los que tenemos ADL, PDDL que serán tratados en el siguiente punto.

2.2.1 Lenguaje de Definición de Dominios de planes PDDL

El lenguaje de Planificación PDDL (*Planning Domain Definition Languaje*), es la extensión más completa de los lenguajes STRIPS, ADL, PDL. Se desarrolló como un estándar para definir dominios y problemas de planificación, su aparición tuvo lugar en 1998 y fue diseñado por el comité de competición de planificación, proporcionando una sintaxis común a los algoritmos para ser utilizado en la Competición Internacional de Planificación[13] [2].

A continuación se describe algunas de las características de este lenguaje, para que los planificadores sean capaces de manejar y elaborar un plan de calidad que se acerque más a la realidad [2] [4]:

- Las acciones del dominio están basadas en el modelo *STRIPS.*
- Sus acciones y objetivos se descomponen creando una jerarquía para tratar el problema.
- Define las formulas lógicas que intervienen en un problema, y las relaciones entre los hechos de un estado.
- Determinan funciones de optimización por medio de las métricas del problema.
- Establece restricciones de seguridad para que se pueda definir los objetivos que se deben cumplir.

[13] Es una competición que se realiza de forma bianual y de forma conjunta con la Conferencia Internacional de Planificación y Scheduling *(ICAPS International Conference on Automated Planning and Scheduling)* y que trata de comparar los planificadores más modernos enfrentándolos a una batería de problemas en diversos dominios descritos en PDDL.
Fuente:http://www.negomobile.es/sites/default/files/data/proyectos/Shop/AplicacionJShop.pdf

- Establece cuantificadores universales.
- Las precondiciones y acciones se asocian desde el inicio hasta el final de la acción para determinar su duración.
- Realiza comparaciones entre expresiones numéricas construidas por operadores numéricos aritméticos.
- Los efectos que causa admiten modificar los valores de las funciones numéricas.
- Se pueden expresar eventos que no se vean dañados por las acciones del plan que sean independientes en ventanas temporales.
- Define restricciones mediante operadores para satisfacer los estados intermedios de los planes.
- El plan se debe cumplir con las restricciones duras y blandas establecidas en el problema, es decir, una restricción dura es aquella que necesariamente debe cumplirse en el plan, mientras que una restricción blanda debe tratar de cumplirse en medida de lo posible.

Considerando todas estas características, es importante decir que este lenguaje puede ser ejecutado en la mayor parte de los planificadores que existen, debido a su calidad en la elaboración de un plan que solucione un problema.

Este tipo de lenguaje también ha evolucionado a lo largo del tiempo, creando extensiones del mismo, como son: PDDL2.1, PDDL2.2, PDD3.0, PDDL3.1[14]. A continuación trataremos los cambios más importantes en cada uno [56] [57]:

- PDDL2.1.- Esta nueva versión aumento la del PDDL crean características como el tratamiento de variables numéricas, así como la representación explicita de tiempo sobre las acciones, con efectos discretos y continuos.
- PDDL2.2.- Incorpora nuevos conceptos en los como los predicados derivados que permiten al planificador razonar sobre conceptos mucho más complejos en el dominio, cuya verdad depende la de los

[14]No existe ningún planificador que soporte la especificación 3.1 completa, sino subconjuntos de ella. Fuente: http://www.lsi.upc.edu/~jvazquez/teaching/iag/transpas/4-PL1-IntroPlanificaci%C3%B3n.pdf

predicados básicos, también incorpora hechos que se determinaran como verdaderos en un instante tiempo.

- PDD3.0.- En esta versión se agrega restricciones que permiten imponer ciertas condiciones sobre la elaboración de los planes válidos y preferencias que llegan a definir objetivos deseados pero no necesarios.
- PDDL3.1.- Básicamente esta versión aplico un lenguaje moderno para obtener una mayor expresividad.

2.2.1.1 Características dominio

En la actualidad PDDL se presenta como un estándar para representar dominios de planificación formados por predicados, funciones y constantes. De igual manera detalla cada una de las características que sean necesarias para que un planificador pueda resolver un problema.

Algunas características importantes para el dominio del problema [13] [57]:

- Contiene acciones, con el detalle de los mismos.
- Cada operador esta descrito por las precondiciones y efectos.
- Las precondiciones determinan las condiciones que se permiten ejecutar una acción.
- Determina cada estado con el tipo de dominio al que pertenece.
- Define los requisitos que plantea el problema con los tipos de datos que tiene.

A continuación se muestra un ejemplo de una sintaxis de código para la definición del dominio del problema tal como se puede ver en la ilustración 8.

```
(define (domain DOMAIN_NAME)
   (:requirements [:strips]  [:equality]  [:typing]  [:ad1])
   (:predicates  (PREDICATE_1_NAME  [?A1  ?A2 … ?AN])
                 (PREDICATE_2_NAME  [?A1  ?A2 … ?AN])  …)
   (:action ACTION_1_NAME
    [:parameters  (?P1 ?P2 … ?PN)]
    [:precondition PRECOND_FORMULA]
    [:efecto EFECT_FORMULA])
(:action ACTION_2_NAME  …)…)
```

Ilustración 8. Definición del Dominio de un problema en lenguaje PDDL [14].

La definición del dominio PDDL contiene, el nombre del dominio (define (domain *DOMAIN_NAME*), la descripción de los predicados*(:predicates (PREDICATE_1_NAME [?A1 ?A2 … ?AN]*) y la descripción de la acción a realizar *(:action ACTION_1_NAME*, cada uno de estos operadores tiene sus precondiciones las cuales se cumplen para la ejecución de la acción[:*precondition PRECOND_FORMULA*] y por lo tanto efectos que tiene la ejecución las precondiciones [:efecto *EFECT_FORMULA*].

Existen varios niveles de expresividad para PDDL, entre los más comunes tenemos [14]:

:*strips* →Expresividad como en STRIPS

:*equality* →El dominio usa el predicado =

:*Typing* →El dominio define tipos de variables

:*adl* →Expresividad extendida: 1) disyunciones y cuantificadores en precondiciones y objetivos 2) Efectos cuantificados y condicionales.

2.2.1.2 Características del problema

Los problemas PDDL contienen una reseña al dominio, lo que le permite dar un argumento al problema, con la descripción del estado inicial y las metas que debe llegar.

A continuación se describe las características [13] [57]:

- Determina el nombre del problema.
- Define el dominio al que pertenece.
- Describe los objetivos mediante ciertas condiciones.
- Las condiciones de un estado debe cumplirse para que se considere como un estado objetivo.
- Detalla cada uno de los objetos con su respectivo tipo.
- Define los predicados que conforman el estado inicial y la meta que tiene que alcanzar el problema.

Como se puede ver en la siguiente ilustración 9 se muestra un ejemplo de código para la definición del problema.

```
(define (problem PROBLEM_NAME)
        (: domain DOMAIN_NAME)
        (: objects OBJ1 OBJ2 … OBJ_N))
        (: init ATOM1 ATOM2 … ATOM_N)
        (: goal CONDITION_FORMULA))
```

Ilustración 9. Definición de un Problema en lenguaje PDDL [14]

2.2.2 Algoritmos de planificación

Uno de los pasos fundamentales en la elaboración de un plan, es la elección de un buen algoritmo de planificación, el cual sea capaz de aprovechar toda la información que tenga el planificador para resolver un problema y obtener planes independientes de dominio.

En la actualidad existe una variedad de algoritmos para la búsqueda, los mismos que han ido evolucionando a lo largo del tiempo, ya que al principio solo podían resolver problemas espontáneos, caso contrario a lo que hoy sucede ya que son capaces de resolver problemas complejos.

Para no tener problemas con la elección de un buen algoritmo es necesario tener en cuenta los siguientes aspectos [6]:

- El espacio que ocupa la búsqueda de estados, planes o grafos para la planificación.

- El tipo de búsqueda según su dirección, hacia adelante, hacia atrás o bidireccional.
- El algoritmo de búsqueda (se trataran a profundidad en el siguiente punto).
- Las técnicas que utilizan, tanto dependientes de dominio como independientes de dominio.
- Las funciones heurísticas que ayudan a determinar la distancia que existe entre un estado y otro.
- Las técnicas de poda ayudando a que el proceso se realice de manera rápida.

2.2.2.1 Algoritmos de búsqueda

Existe una variedad de algoritmos de búsqueda, que se aplican en la elaboración de un plan, cada uno de estos algoritmos decide la generación, ordenación y selección de nodos en un grafo de búsqueda.

2.2.2.1.1 Escala

Los algoritmos por escalada son conocidos también como ascensión a la colina [10], es un procedimiento que trata de elegir en cada paso el estado más conveniente dependiendo, si se desea ascender o descender para mejorar su estado actual, claro está que el algoritmo no elabora un árbol de búsqueda, tan solo observa a su nodo sucesor y almacena el estado actual con mayor valor objetivo, guiado por una heurística y la función de evaluación que genere sucesores. Por ello es de gran ayuda para problemas de optimización (al encontrar soluciones de manera rápida).

A continuación un ejemplo tal como como se ve en la ilustración 10.

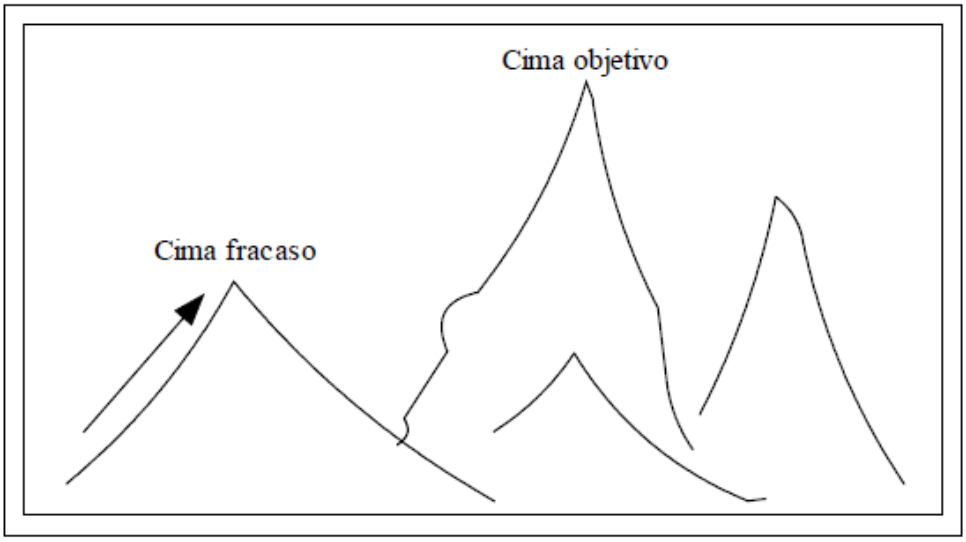


Ilustración 10. Problema de las pequeñas colinas con búsqueda en escala.

De acuerdo al ejemplo de la ilustración 10, podemos observar que el algoritmo consta de un conjunto de colinas en las que se busca el objetivo, es decir cuesta arriba y finaliza cuando alcanza el extremo de la colina superior, dejando a las colinas vecinas con un valor menor. Se utiliza la información que tiene del estado en el cual empieza el problema, para utilizarla en cada paso a lo largo del transcurso de la búsqueda.

Ahora bien el objetivo del algoritmo es llegar a un estado razonablemente mejor que el anterior convirtiéndose de tal manera en una solución sub-optima, después de seguir su recorrido de forma ascendente hasta conseguir el máximo valor local en su función, pero al ser algoritmos locales no garantizan que se encuentre un máximo global.

Tal es el ejemplo TSP (Problema del viajante de comercio) apéndice B1, que elige una perturbación inicial de manera aleatoria para poderla comparar con las posibles soluciones vecinas, las mismas que están formadas por un conjunto de recorridos, que al encontrar una solución dan lugar al nuevo sucesor [18].

Este algoritmo en escala tiene los siguientes métodos [10] [60]:

Métodos Tentativos.- Permite corregir errores, de no elegir la ruta correcta.

Métodos Irrevocables.- No puede corregir si se ha producido algún error en la búsqueda ya que solo puede elegir una ruta, por lo que cada estado genera un nuevo y único estado lo que genera que la lista ABIERTA solo permita asumir un único camino.

Este método da lugar a dos variantes en el momento que se determina qué estado es el sucesor de uno dado [60].

- **La escalada simple.-** Si el camino elegido no es el adecuado, este método no retorna al espacio de estados, el estado se devuelve por lo que ya no puede generar hijos.
- **La escalada por máxima pendiente.-** Calcula el valor heurístico para definir el estado con mejor valor y se compara con el extendido, de resultar mejor que el anterior el estado cambia como expansión.

Este algoritmo puede presentar algunas dificultades al momento de buscar una solución, ya que podrían quedarse varados en estados que no son el objetivo [61].

- Máximos (o mínimos) locales.- probablemente sea el mejor estado en cuanto a sus vecino, pero puede no ser el de otros estados más alejados.
- Mesetas.- Es el lugar en el espacio de estados en el cual todo un conjunto de estados puede tener un mismo valor.
- Crestas.- En el espacio de estados se puede presentar con máximos locales, al ir mejorando el valor heurístico pero sin encontrar ningún objetivo.

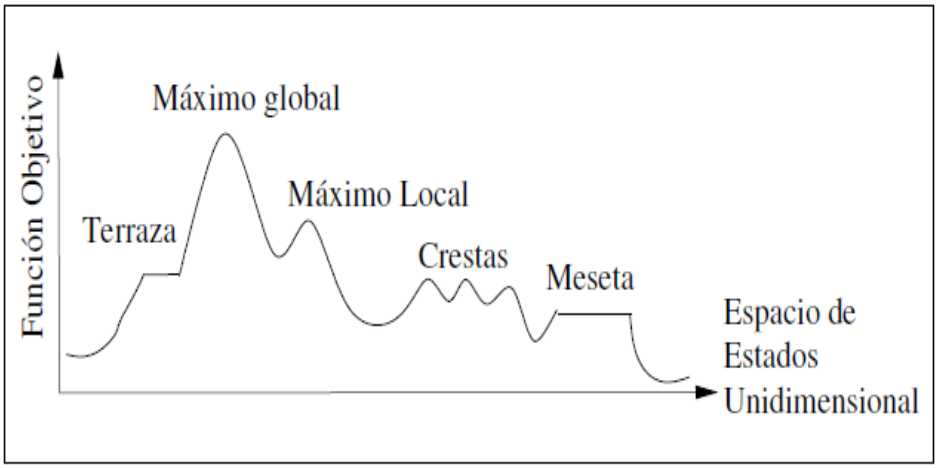


Ilustración 11. Posibles dificultades que se pueden presentar al tratar de encontrar una solución en el proceso de búsqueda [61].

Es importante acotar que existen soluciones para estas desventajas [10]

- Cuando se presenta mesetas, se puede dar un gran salto para evitarlas y seguir con la búsqueda.
- Almacenar el proceso que se realiza para poder regresar a un estado anterior e indagar por otra dirección.

2.2.2.1.2 Enforced Hill-Climbing

Este algoritmo es una sucesión mejorada del algoritmo *Hill-Climbing*, permite realizar una búsqueda mucho más profunda y segura, se utilizan cuando las búsquedas heurísticas no pueden indicar correctamente un camino. Cabe indicar que el algoritmo *Enforced Hill-Climbing* realiza su búsqueda por amplitud para poder comparar estados que se puedan repetir, evitando mesetas y eso sí, basándose siempre en estrategias para alcanzar el estado final, en referencia utiliza el planificador FF (*Hoffmann and Nebel 2001*), el cual poda los estados repetidos para evitar ciclos [20].

En la ilustración 12 se muestra el siguiente ejemplo:

mientras el estado actual no sea solución
buscar un mejor estado usando solo las "helpful actions"
si no se encuentra un mejor estado
buscar un mejor estado usando todas las acciones
si no se encuentra un mejor estado
no se ha encontrado solución
el mejor estado es ahora el actual
devolver el estado actual como solución

Ilustración 12. Representación en lenguaje natural del Algoritmo *Enforced Hill-Climbing* [20].

Entonces el proceso de búsqueda se da desde un estado inicial, luego se escoge un nuevo estado, que tiene que ser mejor que el anterior para obtener el estado actual y de esta manera el valor heurístico va mejorando, además contiene una técnica de poda en la cual solo se selecciona los *helpful* activos[15] [6].

Este proceso se repite varias veces y progresivamente, ya que tiene la ventaja de resolver problemas de máximos locales por su búsqueda en amplitud, expandiendo los nodos capaces de realizar una búsqueda de sucesores de los sucesores hasta llegar al estado meta para cumplir con el objetivo previsto tal como se indica en la siguiente ilustración 13 en la se puede ver el árbol utiliza una cola[16] como estructura de datos, de tal manera, que tiene nodo raíz A, los nodos sucesores de la raíz A y así sucesivamente hasta que se encuentre el nodo meta [62].

[15]Acciones que pertenecen al plan relajado o agregan alguna de las precondiciones necesarias para alcanzar alguna de las acciones del plan relajado. Fuente:

[16]Estructura **FIFO** (First In, First Out) en la que sólo disponemos de dos operaciones: insertar al final de la cola y extraer del principio de la cola. Por tanto, el elemento que entra el último será el último en salir. Fuente: http://blog.vidasconcurrentes.com/programacion/busqueda-en-profundidad-y-busqueda-en-anchura/

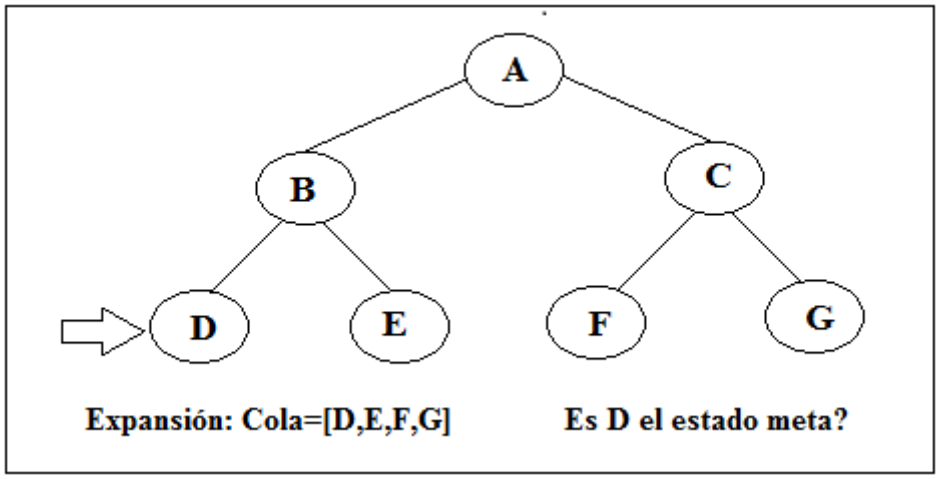


Ilustración 13. Nodos del árbol de búsqueda en Amplitud [22].

El algoritmo tiene una propiedad muy buena, que es poder diferenciar el momento en que se crean los nodos del momento de su evaluación, esto conlleva a que en el momento de la búsqueda en amplitud se localice un sucesor que mejore la heurística, mientras que los sucesores restantes que se encuentren en el mismo nivel ya no serán evaluados ayudando asi en el orden de evaluación de los nodos [6].

2.2.2.1.3 WBFS (*Weighted Best-First Search*)

En este algoritmo se crean los sucesores de cada estado los mismos que están evaluados por una función heurística que indica la distancia que existe entre el estado inicial y el estado final y así se introduce en la lista abierta de los nuevos nodos [6].

La búsqueda se realiza en un árbol de nodos y cuando esta alcanza su mayor profundidad ya no se vuelven a considerar ciertos nodos en las nuevas búsquedas. Es importante actualizar los caminos de los nodos descendientes y los elaborados anteriormente para que puedan encontrar un camino corto hasta alcanzar el objetivo planteado.

2.2.2.1.4 DFBnB (Depth-First Branch and Bound)

En este algoritmo se define su búsqueda primero en profundidad para encontrar una solución, utiliza la función de evaluación para ordenar los

nodos del árbol y eliminar los nodos que sobrepasan el costo del objetivo final y que ya no ofrecen ninguna solución, en la siguiente ilustración 14 se muestra un ejemplo:

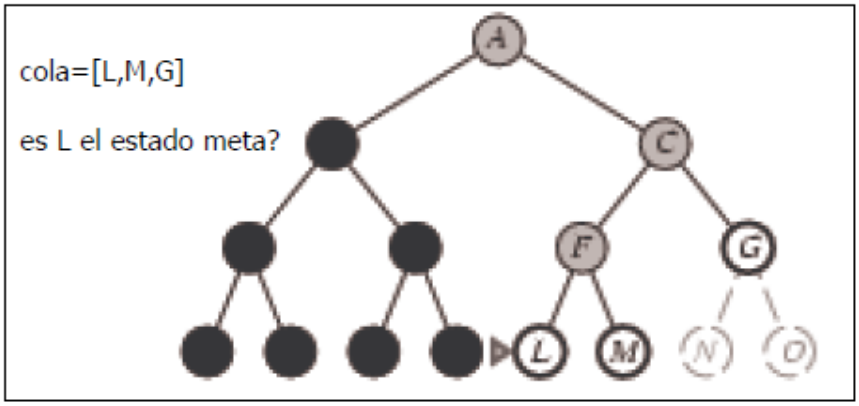


Ilustración 14. Árbol de búsqueda con nodos expandidos en la búsqueda por profundidad *branch and bound* [22].

Este tipo de algoritmo tiene la ventaja *branch and bound* que permite incrementar su calidad en la búsqueda de la solución óptima con la ayuda de las funciones heurísticas, al igual cuando un problema presenta más de una solución otorga una técnica de aprendizaje para mejorar y encontrar varias soluciones, hasta alcanzar el costo óptimo [9].

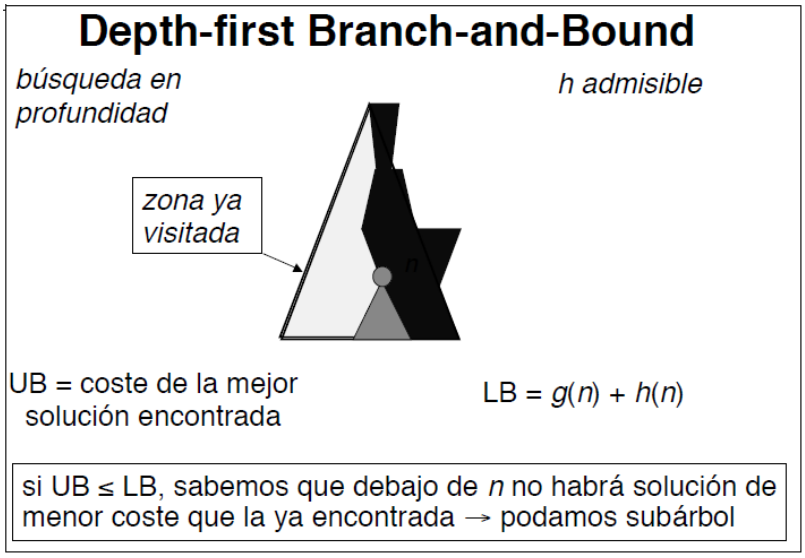


Ilustración 15. Búsqueda Primero en profundidad *Branch and Bound* [9].

En donde [9]:

$h(n)$ Estima el costo desde n hasta encontrar la solución óptima.

$g(n)$ Define el costo desde la raíz hasta el nodo n.

En tanto, si este algoritmo trabaja con funciones admisibles, garantiza que la solución sea óptima.

2.2.3 Planificación heurística

Para la planificación es necesario investigar mecanismos de búsqueda eficaces, que a lo largo del tiempo se ha ido mejorando gracias a contribuciones como *Graphplan*, en el cual el número de acciones es equivalente a la distancia estimada al estado objetivo, el control de búsqueda importante debido a que la mayor parte de los algoritmos utilizan un método de búsqueda para encontrar planes.

Se puede definir varios espacios para búsquedas, existiendo todos estos en tamaños exponenciales lo que puede ocasionar una búsqueda inservible por lo que la planificación debe ser capaz de controlar la búsqueda.

Es entonces la búsqueda está guiada por una función Heurística independiente del dominio, una opción que se da para construir las heurísticas es resolver una versión reducida del problema, conocida como problema relajado que en general consiste en ignorar la lista de borrados en un modelo de acciones, haciendo más sencillo que la resolución del problema original, pero no garantiza la solución óptima.

Existen métodos que ayudan en la resolución de problemas relajados [2]:

- Calcular un plan independiente para cada uno de los estados, para que no existan interacciones entre los diferentes objetivos
- Resolver un problema de manera sub-óptima, como se utiliza en las heurísticas FF.

Así también, se distinguen diferentes relajaciones para la planificación heurística [2]:

- Ignorar los efectos negativos de las acciones.- Es una técnica muy utilizada en cuanto a planificación heurística, cada estado obtiene el

valor de la heurística, estimando el coste de llegar del inicio a la meta en el problema relajado.

- Aproximar un conjunto de hechos mediante subconjuntos.- Se puede estimar su costo al calcular el subconjunto de tamaño m (indica el tamaño de los subconjuntos considerados) de todos los posibles.
- Ignorar determinados hechos.- El espacio para realizar la búsqueda puede resultar mucho más fácil al eliminar ciertos hechos como el estado inicial, final o las acciones del problema.

La planificación heurística se basa en funciones de estimación clasificadas de la siguiente manera [8] [6]:

- **Funciones Admisibles(A*).-** Alcanzan soluciones de búsqueda optimas, pero su cálculo podría ser Np-duro[17], pero este tipo se problemas podría ser tratable al utilizar aproximaciones cercanas en un tiempo real, o construir algoritmos basados en *backtracking*[18]para poder resolver los problemas.
- **Funciones No Admisibles.-** No garantizan que la búsqueda sea óptima, y no resuelven problemas en varios dominios.

Entre los planificadores heurísticos más utilizados se encuentra [11] [34]:

- HSP (*Heuristic Search Planner*).- Fue el primer planificador independiente de dominio, resuelve problemas pequeños dentro de un rango de tiempo, la primera versión utilizo el algoritmo escalada, las versiones posteriores HSP2 hacen uso de otros algoritmos.
- FF (*Fast-Forward*).- Es un planificador desarrollado por *Jorg Homann*, independiente de dominio, realiza la búsqueda en el espacio de estado hacia adelante guiado por las funciones heurísticas.
- AltAlt (*Combining the advantages of Graphplan and heuristic state search*). - Fue desarrollado por *Rao Kambhampati*, es un planificador híbrido con heurísticas muy eficaces ya que utiliza la representación gráfica del algoritmo *Graphplan.*

[17] Es la optimización de un problema completo

[18] Backtracking (o búsqueda atrás) es una estrategia para hacer una búsqueda sistemática a través de todas las configuraciones posibles dentro de un espacio de búsqueda. Fuente: http://web.ing.puc.cl/~jabaier/iic2552/backtracking.pdf

2.2.4.- Técnicas a Utilizar

2.2.4.1 Técnica de Descomposición

En general, esta técnica trata de descomponer un problema en varios sub-problemas, para luego unir todas estas ideas en una sola. La técnica tiene como objetivo hacer más fácil la labor de construir y buscar la solución a un problema.

Es fundamental destacar algunas ideas de descomposición para llevar a cabo este proceso, entre las mismas tenemos [2]:

- Clasificar y ordenar los objetivos para determinar el tipo de objetivo que tiene cada estado.
- La descomposición de objetivos es una técnica utilizada por el planificador SGPlan, ya que estudia las amenazas que se pueda presentar y por lo tanto afectar los objetivos, al elaborar más de un plan para cada objetivo se realiza un análisis mucho más profundo.
- Descomponer un problema en distintos niveles se denomina planificación jerárquica (este punto será tratado a profundidad en el Capítulo 4).

2.2.4.2 Técnica Lookahead

Su aparición tuvo lugar en el año 2004 con el planificador *YASHP*, en el cual se introdujeron técnicas muy importantes que ayudan en la eficiencia de ciertos dominios de nodos, esta técnica tiene la ventaja de activar o desactivar acciones en el momento de elaborar un proceso para un plan relajado.

Así mismo, se introdujo en las técnicas de búsquedas heurísticas, primero el mejor, el cual ordenaba la lista de los valores de cada nodo de un árbol para encontrar el nodo más significativo, ahorrando evaluaciones a la función heurística, ya que su función está definida como ayuda en la toma de acciones de un plan ya elaborado, su proceso se repite de manera iterativa, hasta cuando ya no se encuentren acciones que sean aplicables finaliza [6].

2.2.4.3 Técnicas Híbridas de Descomposición

Estas técnicas son ciertamente más complejas que las ya mencionadas pero incluyen buenos resultados, que es precisamente lo que un planificador busca en la resolución de un problema.

Entre los planificadores que utilizan este tipo de técnicas se pueden destacar los siguientes [2] [63]:

- Mips.- Es un planificador que da lugar a planes temporales, utiliza el algoritmo de búsqueda ascenso a la colina.

  Este algoritmo trabaja con:

  - Efectos condicionales
  - Precondiciones negadas
  - Expresiones numéricas
  - Acciones durativas

- LPG.- Es un planificador que tiene como fundamento la búsqueda local y los grafos de planificación, en cada paso realiza diversas modificaciones para transformar un plan parcial en otro plan mejor, para poder incrementar la calidad del proceso de búsqueda, su evaluación se realiza mediante una función que utiliza varias heurísticas con un procedimiento para problemas de satisfactibilidad[19].
- AltAlt.- Es un planificador que guía a los planificadores heurísticos[20].

[19] Utilización de las formulas booleanas proposicionales como un lenguaje de programación con restricciones para solucionar problemas NP-completos.
Fuente: http://polar.lsi.uned.es/revista/index.php/ia/article/viewFile/374/359

[20] Descrito en el punto 2.2.3 Planificación heurística.

CAPITULO III

3 Aprendizaje aplicado a la planificación

Las metodologías o técnicas de Aprendizaje Automático (AA) a lo largo del tiempo han complementado o asistido a la Planificación Automática (PA), ayudando y mejorando el rendimiento de un sistema, aumentado sus prestaciones, ya que las metodologías en cuestión adquieren conocimiento mediante la experiencia. Con esto se pretende lograr que los sistemas que apliquen estas técnicas incrementen su eficacia, ya que la PA se ha convertido, desde el ámbito informático en una tarea con un alto grado de complejidad [6].

Resumiendo el aprendizaje en la planificación busca o pretende aprender el modelo del dominio para así obtener una mejor y optima representación.

Las técnicas de aprendizaje automático se dividen de muchas maneras, por lo cual nos enfocaremos en las más conocidas e importantes [6] [23]:

- **Aprendizaje inductivo:** Se pretende una función a partir de una serie de ejemplos que permiten identificar futuras instancias, dichos ejemplos pueden ser positivos o negativos dependiendo de si i satisfacen la función o no, respectivamente. El factor clave aquí es la utilización del lenguaje, que es empleado para representar cada instancia, el cual se representa o toma forma de tupla con atributos y valores.
- **Aprendizaje analítico:** Busca el aprendizaje más eficiente en una función mediante la explicación, análisis y profundización de un único ejemplo de una función, requiriendo un conocimiento correcto, exacto y completo del dominio.
- **Aprendizaje analítico- inductivo:** Tanto en los métodos inductivos y analíticos existen problemas por lo que algunos investigadores, para cubrir y solucionar inconvenientes fusionaron los métodos teniendo como resultado un sistema multi-estrategia, en el que el conocimiento obtenido por razonamiento analítico es pulido mediante una serie de ejemplos.

3.1 Aprendizaje del Modelo del Dominio

Previo a la planificación, el objetivo del aprendizaje del modelo del dominio, es enriquecer o nutrir el conocimiento de un determinado dominio, claro está, este debe hacerse mediante un análisis. Actualmente existen herramientas de análisis del dominio como por ejemplo el TIM (*FoxandLong*, 1998), donde se descubren estados finitos que representan los estados por los que pueden pasar una instancia en un problema de planificación, encontrando simetrías con lo cual conlleva a podar el árbol de búsqueda del planificador, de esta manera se reducían los esquemas de instanciación delos operadores (*GRAPHPLAN*). Otras herramientas se enfocan en refinar o entender la definición de los operadores del dominio [6]. Por otro lado RIFO es un método que detecta información irrelevante en planificadores basados en espacio de estados, considerando dicha información como innecesaria ya que a partir de esta, no se puede encontrar un plan que establezca la solución, en un pre-proceso inicial antes de que el sistema empiece a resolver el problema , la información irrelevante es eliminada.[23].

En definitiva el aprendizaje del modelo del dominio tiene como objetivo principal *"**completar los procesos originados por teorías incompletas mediante el establecimiento de suposiciones válidas***" [24].

3.2 Aprendizaje de conocimiento de control

Consiste en adquirir conocimiento dependiendo del dominio, complementando el proceso de búsqueda, identificando y dando prioridad a los objetivos u oportunidades para aprender y seguidamente se debe representar dicho aprendizaje. El proceso de aprendizaje puede ejecutarse durante la planificación o durante la obtención de un plan valido [6].

HAMLET es un sistema que combina 2 tipos de aprendizajes, los cuales son el inductivo y deductivo (analítico), convirtiéndose en un sistema multi-estrategia, el cual aprende automáticamente reglas de control para el planificador *PRODIGY* [3], teniendo como entradas una teoría del dominio y

un conjunto de problemas de entrenamiento. Estos problemas son resueltos por *PRODIGY*, que además guarda el árbol de búsqueda formado hasta encontrar una solución; y una medida de calidad o métrica que hace referencia a la longitud del plan, el tiempo que se demora en encontrar la solución, el coste económico del plan, o incluso combinaciones de más criterios [23]. Estas medidas son utilizadas para evaluar la calidad de las soluciones encontradas por *PRODIGY*, maximizando dicha calidad en base al control de conocimiento generado [23].

HAMLET tiene en el módulo de aprendizaje deductivo varios parámetros como muestra la ilustración 16, los más importantes son [3]:

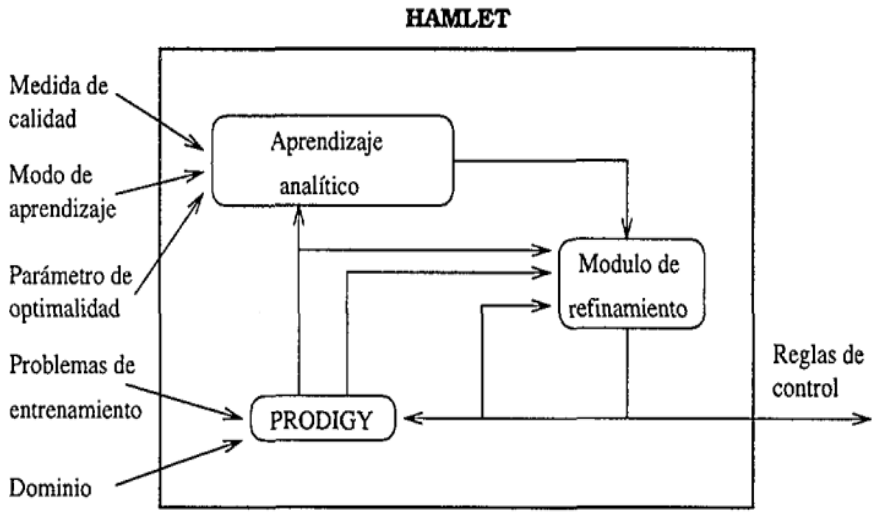


Ilustración 16. Arquitectura de Hamlet que combina aprendizaje deductivo e inductivo, el cual adquiere conocimiento en base al dominio [3].

- **Medida de calidad:** como ya se explicó anteriormente, evalúa la calidad de soluciones, por ejemplo, pueden ser medidas de calidad, el tiempo tardado en encontrar un plan, longitud y costes de un plan.
- **Modo de aprendizaje:** permite que el aprendizaje se haga sobre todos los puntos de decisión del camino, la solución durante la resolución del problema o únicamente sobre las decisiones correctas del planificador.

- **Parámetro de optimalidad:** Hamlet permite empezar el aprendizaje sobre un árbol de búsqueda que aún no se ha generado en su totalidad, ya que por lo general el problema debe resolverse y generar un árbol de búsqueda completado a un 100%.

En la ilustración 17 se describe un ejemplo de regla de control aprendida por Hamlet en el dominio logistics (apéndice A1), para la selección del operador *unload-airplane*. La condición dice que si la meta es mantener un paquete en una ciudad <*location1*>, y mencionado paquete se encuentra dentro de un avión, el planificador debería aplicar o ejecutar el operador unload-airplane. Hamlet induce esta regla general a partir de dos reglas específicas aprendidas en el módulo deductivo de dos casos de entrenamiento diferentes [23].

En el primer caso el avión se encuentra en un aeropuerto distinto de <*location1*>, y la primera regla aprendida tiene como precondición un *true-in-state* adicional, que indica la localización del avión.

En el segundo caso el avión si se encuentra en el aeropuerto de <*location1*>, añadiéndose un meta-predicado *true-in-state*. *HAMLET* mediante su módulo deductivo detecta que estas dos reglas son absorbidas por la regla más general, lo que trae como consecuencia que se quiten las dos reglas más específicas [23].

```
(control-rule  select-operators-unload-airplane
    If (and (current-goal   (at <package><location1>))
            (true-in-state  (inside  <package><airplane>))
            (different-vars-p)
            (type-of-object  <package>  package)
            (type-of-object  <airplane>  airplane)
     (type-of-object  <lacation>   airport) ) )
(then select operator unload - airplane)
```

Ilustración 17. Ejemplo donde Hamlet aprende una regla de control para la selección del operador *unload-airplane* en el dominio *logistics* (apéndice A1) [23].

Hasta el momento se ha revisado a breves rasgos el sistema Hamlet y como se estructura su arquitectura, pero debemos volver al conocimiento de control, donde se consideran objetos de aprendizajes a [6]:

3.2.1 Puntos de decisión

En la búsqueda de espacios se puede escoger sobre qué instancia de operador elegir y qué amenazas solucionar, claro está, dependiendo de qué algoritmo se aplique y del modelo de planificación.

3.2.2 Puntos de retroceso

Depende de los nodos de fallo que se encuentren en el árbol, ya que así el algoritmo aplicado aprenderá de los puntos de retroceso.

3.2.3 Funciones de evaluación

El cual busca el ordenamiento de los nodos existentes en el árbol, por medio del refinamiento de funciones de evaluación.

3.2.4 Episodios de planificación

Se pretende el aprendizaje parcial o totalmente en conjunto el proceso de planificación, esto puede ser una solución completa, macro –operadores, transición de estado, y secuencias a partir del árbol.

En la ilustración 18 se ejemplifica un árbol de búsqueda que tiene puntos de decisión de donde se pueden aprender reglas, nodos de éxito, de fallo, desconocidos y de abandono, este último que se refiere a nodos que no son ni de éxito ni de fallo, teniendo al menos un sucesor [23].

A continuación se presenta un ejemplo, que es representada por la ilustración 18:

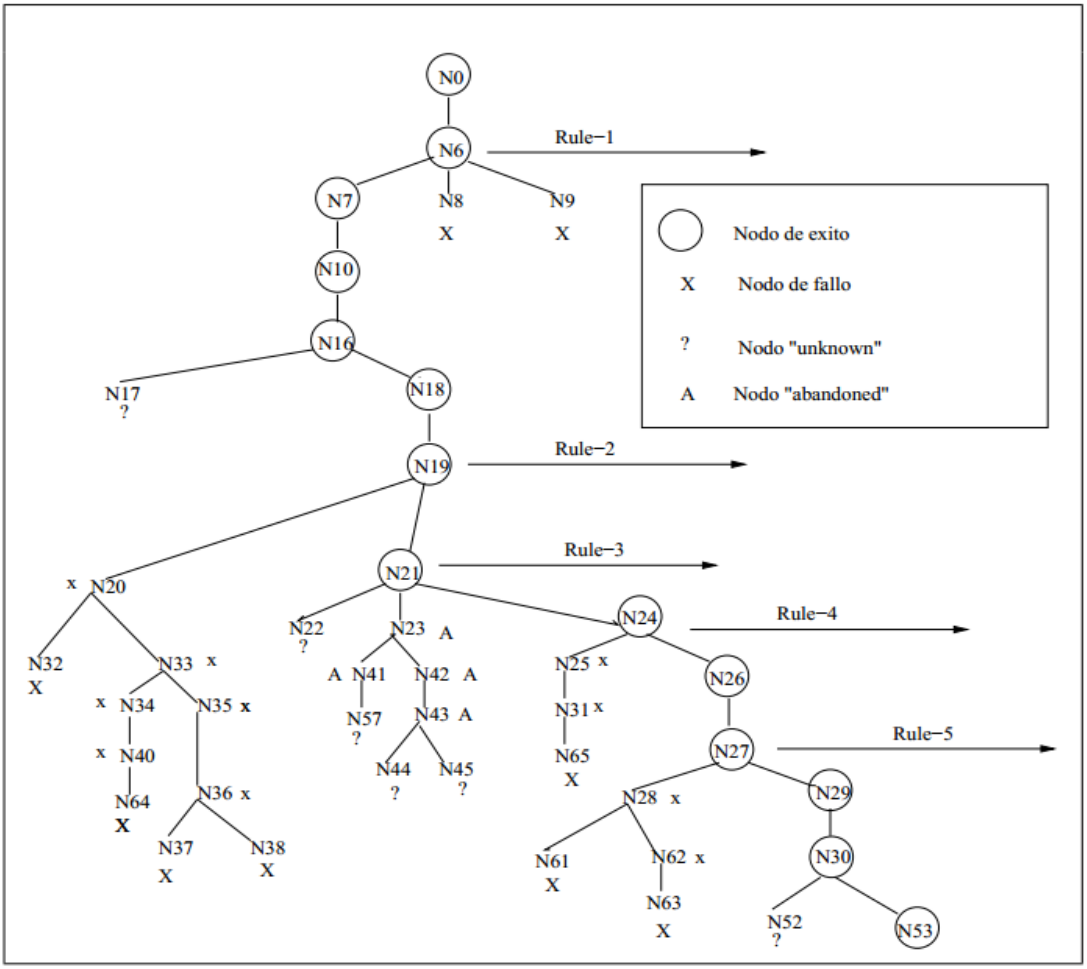


Ilustración 18. Árbol de búsqueda, con puntos de decisión, de donde pueden aprender reglas, junto con sus respectivos nodos [23].

3.2.4.1 Macro – operadores

Los macro-operadores son secuencias de operadores de planificación que pueden ser reutilizados con la finalidad de encontrar una solución más rápida [23], teniendo sus propias precondiciones (*subconjunto de las precondiciones de los operadores empleados en la búsqueda*) [25], reduciendo la profundidad del árbol de búsqueda. Esto conlleva un avance más acelerado hacia las metas, aunque existe el inconveniente de que si bien reduce la profundidad del árbol, aumenta el factor de ramificación debido a que añade o incrementa operadores del dominio [6].

En la ilustración19 se describe un ejemplo de macro-operadores, en el dominio de mundo de bloques (apéndice A2), del cual se intenta llevar y apilar bloques en una mesa por medio de un brazo, donde P es un conjunto de literales (los cuales describen como debe ser el estado del sistema para poder aplicar el operador) y A que es conjunto de literales que se añadirán a la representación del estado del sistema una vez ejecutado el operador [25].

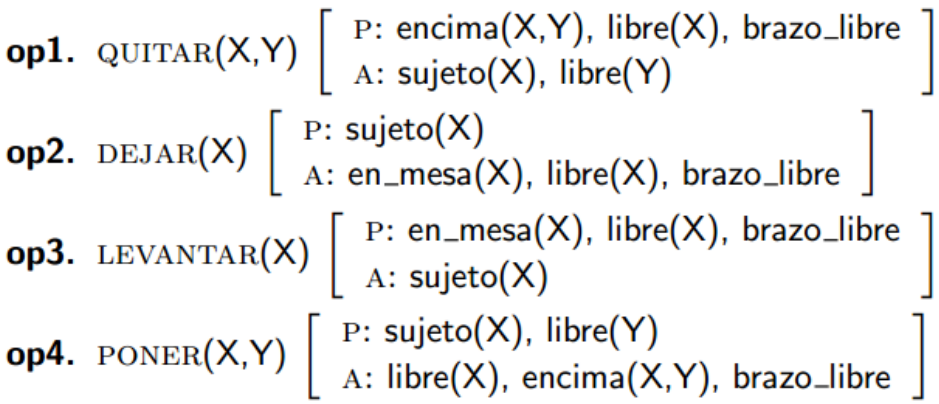


Ilustración 19. Ejemplo de macro-operadores en el dominio de bloques (apéndice A2) [25]

CAPÍTULO IV

4.1. Planificación basada en Casos

La planificación basada en casos tuvo sus inicios junto con la planificación basada en el razonamiento, su aplicación nació en distintas áreas como la robótica y uno de sus primeros planificadores fue *CHEFF*[21][26].

Básicamente la planificación basada en casos, evita el trabajo de elaborar planes desde su principio, ya que permite almacenar en memoria los planes previamente creados para ser reutilizados posteriormente. Estos planes se adaptan a cada situación según las condiciones y necesidades que presente un problema.

Además, aporta conocimiento al proceso de planificación y evita la pérdida de tiempo para ganar en cuanto a costos y potencial heurístico en problemas complejos con soluciones mucho más eficientes [27].

Esta planificación ha sido utilizada con grandes ventajas en la planificación automática, al examinar de forma inteligente cada una de las secuencias de acciones, hasta lograr cumplir con el objetivo, así también se ha desarrollado en varios ámbitos como son: el diagnóstico de enfermedades, diseño de software, juegos en tiempo real, etc. [6]

Se describen dos tipos de sistemas CBP [28]:

- Según la estructura que presenten los planes, pueden ser lineales o jerárquicos.
- Según los que hacen uso de problemas parciales para crear un problema completo (estos puntos se profundizan más adelante).

4.1.1. Razonamiento Basado en Casos

Aportan mucho a los problemas de planificación, ya que los planes que ya han sido elaborados y a su vez utilizados en la solución de otros problemas

[21]Recuperaba de la memoria de casos, el caso más parecido en función de las características dadas en las metas y luego adaptaba la receta a la nueva situación, primero instanciando el plan con los nuevos objetos, y luego reparando el plan en función de anotaciones que se incluían en el caso para anticipar los fallos. Fuente Razonamiento Basado en Casos Aplicado a la Planificación Heurística.

son almacenados en una memoria de casos (MC) [29], para luego poder ser reutilizados.

Como se observa en la ilustración 20, tenemos un ejemplo de la estructura de la memoria de casos, que está compuesta por la descripción del problema, su solución y los resultados obtenidos [29].

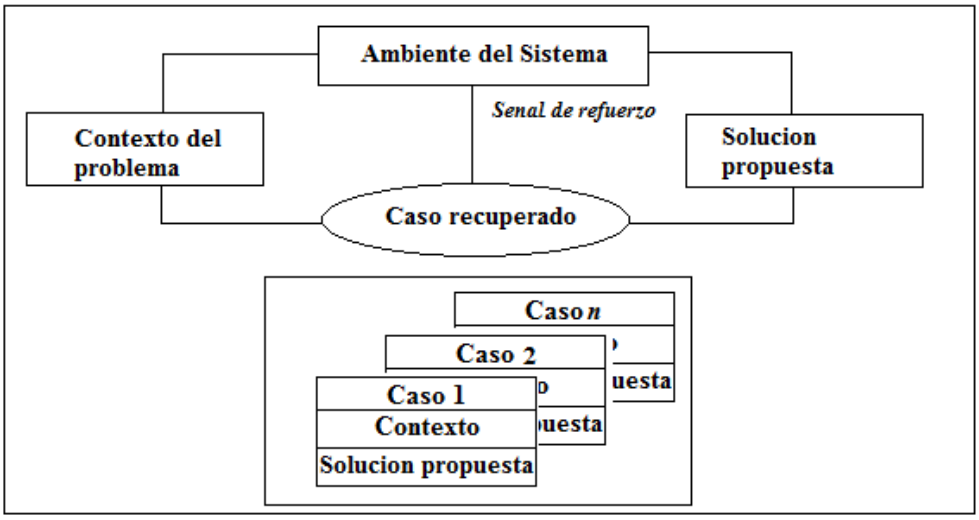


Ilustración 20. Estructura de la menoría de casos - Ciclo de CBR [29].

Existen distintas fases que hacen posible que el razonamiento basado en casos para que colabore en el nuevo proceso de planificación [30]:

- **Recuperación.-** Se analizan las diferentes características del problema para comparar la similitud que existe con las características ya almacenadas de casos anteriores, para poder hacer uso de las que sean útiles y similares y asi aplicarlas nuevamente.
- **Adaptación.-** Los casos que se han determinado como similares y útiles deben ser adaptados a cada necesidad.
- **Revisión.-** Luego de adaptar cada caso se debe realizar una revisión, que determine si la solución que se planteó es la apropiada, en caso de presentarse algún error se repara el fallo para evitar errores posteriormente.
- **Almacenamiento.-** Se debe almacenar la solución que se dio al problema junto con sus resultados, todos estos datos se guardarán en

la memoria de casos para ser utilizados en nuevos problemas en un futuro.

Como se aprecia en la ilustración 21, tenemos un ejemplo de CBR en el que se indica las necesidades X de un estudiante y las necesidades Y de otro estudiante. Lo que se busca es encontrar las necesidades semejantes entre los dos [31]:

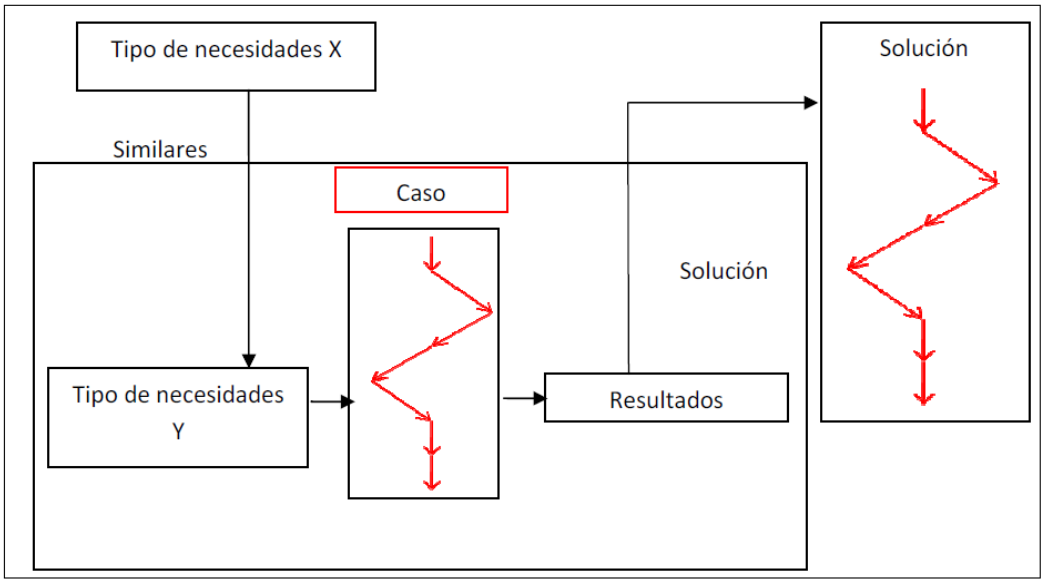


Ilustración 21. Selección de un plan utilizando el razonamiento basado en casos para encontrar necesidades semejantes entre dos estudiantes [31].

4.1.2 Darmok

Es un sistema de planificación basado en casos para juegos de estrategia en tiempo real, funciona de la siguiente manera: Un jugador inicia el juego e indica al sistema la forma como lo tiene que hacer, a cada paso que hace el jugador, el sistema lo guarda como indicación .Estas indicaciones son almacenadas a manera de planes en la base de casos y luego el sistema elabora un plan jerárquico que será utilizado por *Darmok*[22], que tiene como prioridad ganar el juego [43].

[22] Puede jugar a cualquier juego de competición entre adversarios.

A este sistema se agrega una versión mejorada, llamada el ciclo OLCBR que se define en las siguientes fases: [43].

- Fase de expansión.- Como ya se mencionó, el ciclo CBR elabora un plan jerárquico, es decir, crea sub-problemas de un plan y se toma la solución actual para analizar los sub-problemas que se encuentran ya abiertos. Dichos sub-problemas son enviados a la fase de recuperación para finalmente resolver el problema, es así como se realiza el proceso de expansión.
- Fase de ejecución- Esta fase es la encargada de ejecutar y actualizar el plan.

4.2. Planificación basada en el espacio de Estados

Como ya hemos visto en los capítulos anteriores la planificación es un proceso de búsqueda, y es precisamente la búsqueda en un espacio de estados la manera más relajada de realizarla.

La búsqueda se muestra en forma de un árbol compuesto por distintos nodos, en donde cada nodo representa un estado, tal como se muestra en la ilustración 22 [22].

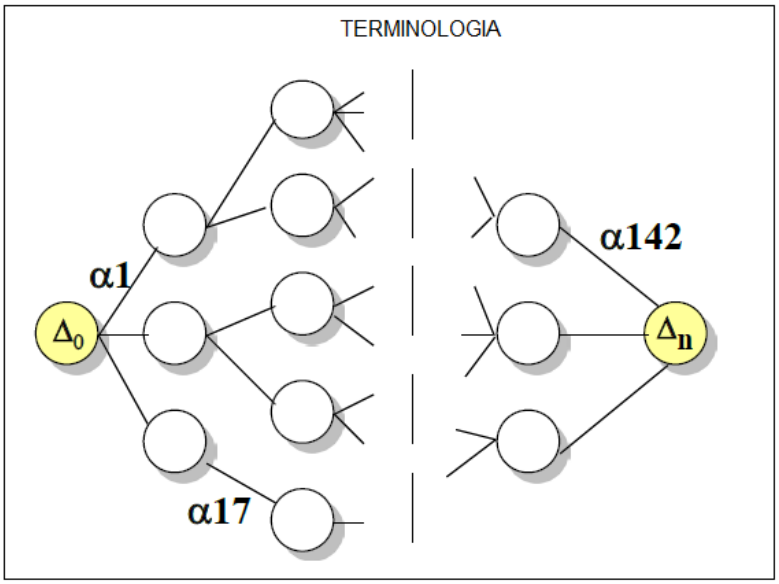


Ilustración 22. Distintos nodos que forman un árbol - Búsqueda en un espacio de estados [12].

Entonces, un espacio de estados es el conjunto de todos los estados posibles junto con sus operadores, secuencia de acciones y las diferentes rutas de búsqueda que existen entre ellos, desde su inicio hasta encontrar la solución en su estado objetivo.

Así como se utilizaba en el lenguaje de representación *STRIPS*, esta planificación tiene la finalidad de elaborar un plan consistente, tal como se puede ver la ilustración 23, en la que el espacio de estados forma un grafo, los nodos representan los estados y los arcos representan las acciones. A partir de esto, se determina el estado en donde empieza la búsqueda, las acciones con la ruta a seguir y el costo total de llegar alcanzar la solución [22].

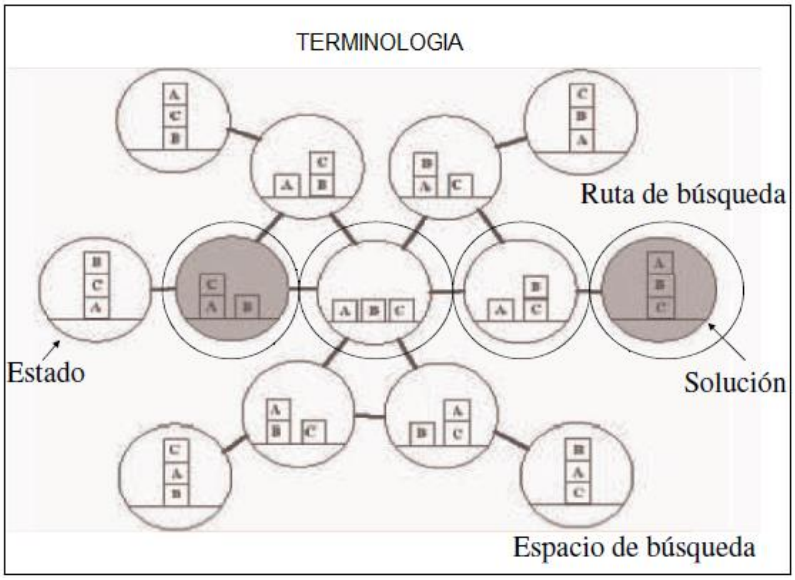


Ilustración 23. Terminología de búsqueda con planificación basada en casos [22].

Para resolver un problema con la búsqueda en un espacio de estados, se debe hacer uso de un algoritmo de búsqueda con sus respectivas funciones heurísticas independientes de dominio (ver capitulo II), que sean capaces de ayudar al planificador a construir planes ordenados.

La búsqueda se puede realizar en dos direcciones, tanto hacia adelante como hacia atrás [11].

4.2.1. Búsqueda hacia adelante

Conocida también como progresión, ya que establece y mantiene una dirección de avance, dado que examina una secuencia de acciones que permitan llegar al objetivo, crea para cada estado un predecesor.

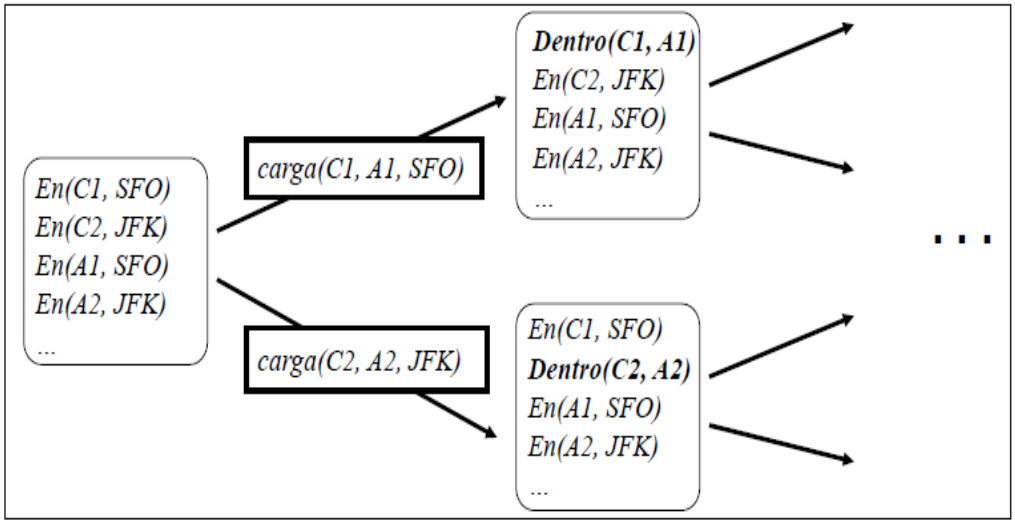


Ilustración 24. Búsqueda en el espacio de estados hacia adelante (Progresión) [12].

Es fundamental definir los siguientes puntos [12]:

- **El estado inicial.-** Indica en donde comienza el proceso de búsqueda en el espacio de estados.
- **Definir las acciones.-** permiten cambiar de estado siempre que se cumpla con las precondiciones.
- **Definir los objetivos**.- Para determinar si la búsqueda se realizó de manera exitosa.
- **El costo**.- El costo final que tiene para alcanzar un objetivo, que se obtiene con suma total de cada acción que forma un plan.

4.2.2. Búsqueda hacia atrás

La búsqueda hacia atrás es conocida también como búsqueda regresiva y tiene cierta ventaja sobre la progresiva, ya que necesita un número menor de acciones para ser aplicadas en el proceso de búsqueda. Esto se logra ya que

considera solo las acciones relevantes[23] y consistentes que permiten definir los estados predecesores, los mismos que indican desde donde inicia la acción para alcanzar el objetivo [64].

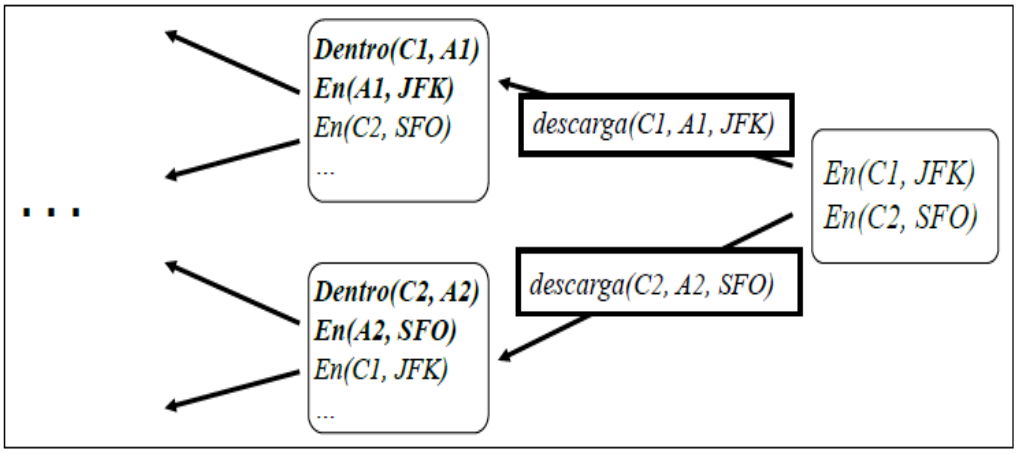


Ilustración 25. Búsqueda en el espacio de estados hacia atrás (Regresión) [12].

En este caso se empieza por el estado objetivo, luego se determina cada uno de sus operadores, precondiciones y literales por el que está constituido [34], para concluir cuando se produce un conjunto formado por todos los sub-objetivos que encuentran una solución en el estado inicial.

En la ilustración 26 se observa el esquema básico de cómo funciona la búsqueda hacia atrás, su objetivo está en actualizar siempre el estado final hasta que alcance el inicial, su proceso parte de un plan vacío, el cual se va elaborando con cada iteración del algoritmo. Por ello, se debe aplicar un bucle para obtener el conjunto de acciones, las mismas que pueden ser relevantes y luego se requiere agregarlas directamente al plan vacío, caso contrario se devuelve un fallo [4].

[23]Una acción es relevante a un objetivo si su efecto agrega uno de los literales del objetivo.

ESQUEMA BASICO DE LA BUSQUEDA HACIA ATRAS

1: $\Pi \leftarrow$ Plan vacío
2: **bucle**
3: **si** $\mathcal{G} \in \mathcal{I}$ **entonces**
4: Devuelve Π *{Éxito}*
5: **fin si**
6: *Relevante* $\leftarrow \{\mathtt{a} \mid \mathtt{a}$ es una acción relevante para $\mathcal{G}\}$
7: **si** *Relevante* $= \emptyset$ **entonces**
8: Devuelve fallo *{No existe plan}*
9: **fin si**
10: Seleccionar no determinísticamente y extraer $\mathtt{a} \in$ *Relevante*
11: $\Pi \leftarrow \mathtt{a} \cdot \Pi$
12: $\mathcal{G} = \mathcal{G} - \mathrm{Efe}^{+}(\mathtt{a}) \bigcup \mathrm{Pre}(\mathtt{a})$
13: **fin bucle**

Ilustración 26. Algoritmo del esquema básico de la búsqueda en un espacio de estados hacia atrás [4].

En donde:

$\boldsymbol{G}$ →Objetivo del problema

$\boldsymbol{I}$ →Conjunto de sub-objetivos

$\boldsymbol{II}$ →Plan vacío

El proceso se inicia desde el plan vacío II, el cual se va construyendo continuamente en cada iteración del algoritmo y a su vez se va actualizando G hasta llegar a I. Por cada iteración que realice el bucle, se obtiene una acción relevante, la misma que busca conseguir algún objetivo en G, de ser así se actualiza el plan, caso contrario devuelve fallo (el plan no existe) [4].

Entre los planificadores que utilizan este tipo de búsqueda tenemos los siguientes [33]:

Progresión	Regresión
SOAR	*STRIPS*
FF	*UCPOP*
TLPLAN	*PRODIGY*

Tabla 1. Planificadores para la búsqueda en el espacio de estados progresión y Regresión [33].

4.2.3 Heurísticas para la búsqueda en un espacio de estados

Como ya hemos visto, la idea principal de las heurísticas es alcanzar los objetivos con la ejecución de las acciones, cabe indicar que se debe determinar el número de acciones que se necesiten para lograr los objetivos, para lo que se estos enfoques [46]:

- **Problema Aproximado.-** Obtiene una heurística admisible al analizar y sintetizar el problema con los detalles iniciales y a su vez va eliminando listas vacías.
- **Sub-objetivos independientes.-** Divide los objetivos en sub-objetivos, los cuales pueden ser optimistas si tiene interacciones negativas entre los sub-planes de cada sub-objetivo o pesimistas cuando sus sub-planes obtienen acciones redundantes.

  Se pueden considerar heurísticas por las acciones que presentan sus interacciones positivas, que van eliminando los efectos negativos y, por lo tanto, no hay la necesidad de estar pendiente de las interacciones negativas entre los sub-planes, ya que no ocasionan que una acción pueda eliminar los literales alcanzados por otra acción. Es importante mencionar que estas heurísticas pueden ser utilizadas tanto en la búsqueda hacia adelante como en la búsqueda hacia atrás [46].

4.3. Planificación Numérica

Planificación temporal: tiempo+recursos

En los problemas de planificación, un factor clave es el tiempo que toma realizar las acciones, ya que no todas tienen la misma duración. Por ejemplo, en un problema de dominio de *logistic (apéndice A1)* la acción *load-airplane* tiene un tiempo de duración diferente a la acción *load-truck*, debido a factores de velocidad, distancia, y el medio de transporte (avión/camión), etc. Por ende, todas las acciones no tienen el mismo tiempo de duración, siendo el planificador el encargado de gestionar o administrar las características de tiempo directamente. Es por esto que la planificación temporal se enfoca en minimizar el tiempo de un plan, sin considerar el número de acciones [4] [35].

4.3.1 Planificador TPSYS

Analizaremos el planificador temporal *TPSYS*, el mismo que está basado en *Graphplan* y TGP (Temporal *Graphplan*) y está formado por tres etapas [35]:

- **Primera etapa:** En esta etapa se calculan las relaciones de exclusión estáticas entre acciones que parten a partir de la definición del dominio.
- **Segunda etapa:** Principalmente radica en la expansión incremental de un grafo temporal, etiquetando las acciones con el tiempo en las que van apareciendo, generando en cada nivel acciones que consten de precondiciones y que no sean excluyentes en el mismo nivel, teniendo en cuenta que depende de la duración de la acción. La expansión del grafo finalizará cuando todos los objetivos del problema se encuentren en un nivel temporal de proposiciones, y sin ser excluyentes entre sí.
- **Tercera etapa**: Mediante un flujo de acciones extrae un plan ejecutable, estudiando cada una de las acciones que se genera a partir

del objetivo, eligiendo aquella acción que no sea excluyente con acciones planificadas, así se obtendrá un plan con mínima duración.

4.3.1.1 Modelo de planificación

Para representar el modelo de planificación se especifica la tupla $\{I_s, A, F_s R D_{max}\}$, donde I_s y F_s representan la situación inicial y final, respectivamente; R^+ que representa el tiempo; *A* indica el conjunto de acciones que poseen duración y que pueden requerir cualquier recurso *R*; finalmente D_{max}, que representa el tiempo máximo del plan permitido por el usuario [35].

4.3.1.2 Naturaleza de los recursos

Se refiere a cualquier objeto, que por lo general es limitado y necesario para la ejecución de una acción, que conlleva a la restricción del plan, debiendo el usuario expresar la disponibilidad de recursos, y representarla en el dominio [35].

Como ya se explicó anteriormente, *R* representa cualquier recurso que puede agruparse de acuerdo a subclases $R(R_{tipo} \subseteq R)$.

Como ejemplo, si en nuestro dominio *logistics (apéndice A1)*, tenemos 5 camiones y 2 aviones, habría dos tipos de recursos: R_{camion} y $R_{avion}/R = R_{camion} \cup R_{camion}$. Asimismo, existen recursos intercambiables $R_{intercambiable} \subseteq R_{tipo}$, pero estos dependen de la situación inicial, ya que si no es la misma no se podrá utilizar de forma indistinta. Para un mayor entendimiento sobre recursos intercambiables, se pone el caso de que si en el problema anterior hubieran solo aviones en X localidad e Y localidad, habrían 2 tipos de recursos compartidos R_{camion_X} y R_{camion_Y}, los cuales pertenecerían a R_{camion}, [1].

4.3.2 Arquitectura TPSYS

Anteriormente se explicó que TPSYS constaba de tres etapas (lo que se muestra en la ilustración 27), por ello, después de la primera etapa de proceso, la segunda y tercera etapa se intercalan, hasta obtener el plan de duración óptima, o la duración del plan supere a D_{max} [35].

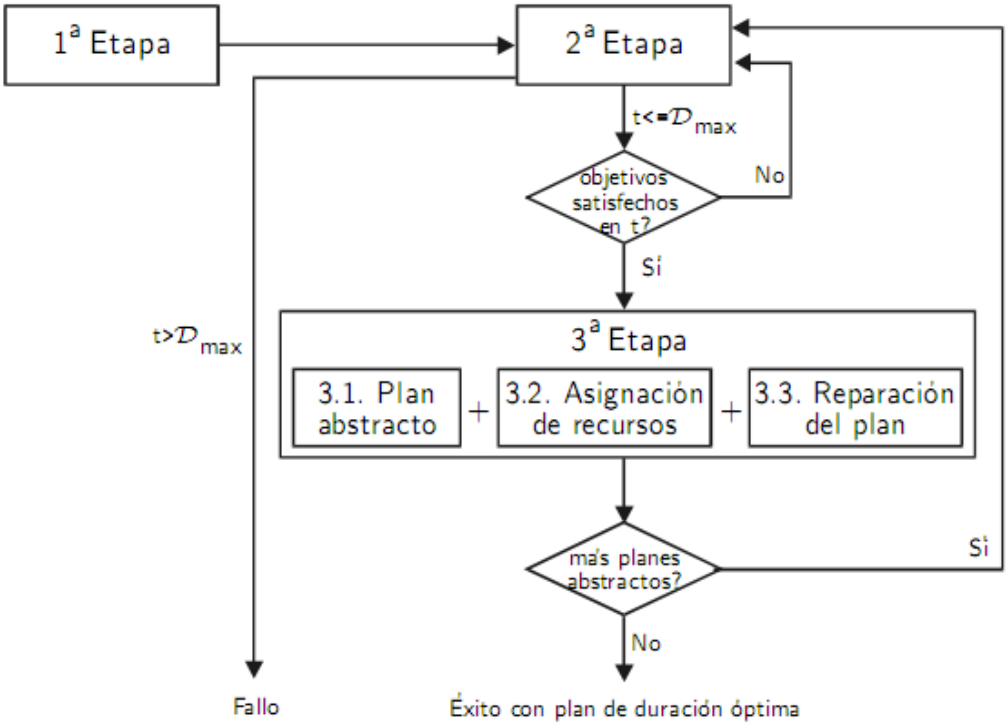


Ilustración 27. Diagrama con las 3 etapas de TPSYS, para definir su arquitectura [35].

Detallaremos un ejemplo sencillo, basado en el dominio *logistic*, el cual nos ayudará a entender las tres etapas del sistema. Tendremos tres objetos ($O1, O2, O3$) que serán transportados desde la localidad X a la localidad Y, para ello se ayudarán acciones ya definidas, que están representadas en la ilustración 28. Se añade tres acciones análogas a embarcar, volar y desembarcar, las cuales utilizan un recurso de tipo avión R_{avion} con duraciones 50, 40 y 80 respectivamente. En la situación inicial se contará con un solo camión en la localidad X y un solo avión en la localidad Z, lo que implica ejecutar la acción volar de duración 100 ya que se necesita movilizar el avión de Y a X [35].

(:action cargar	*(:action mover*	*(:action descargar*
: parameters ()	*: parameters ()*	*: parameters ()*
: precondition ()	*: precondition ()*	*: precondition*
():effect ()	*: effect ()*	*: effect ()*
: duration 50	*: duration 100*	*: duration 80*
*: lock (**Rcamion**)*	*: lock*	*: lock*
: unlock ()	*: unlock ()*	*: unlock (**Rcamion**)*
*: use (**Rcamion**))*	*: use (**Rcamion**))*	*: use (**Rcamion**))*

Ilustración 28. Ejemplo basado en el dominio *logistic* - Parte de un dominio de transporte [35].

4.3.2.1 Primera etapa: Abstracción de recursos

En esta etapa se logra la consecución de los objetivos por medio de recursos (**?recurso**)ilimitados, en el ejemplo planteado el O1 solo podrá ser transportado por la variable camión ?CM1, o por la variable avión ?AV1, mientras tanto los objetos O2, y O3 solo podrán ser transportados por las variables ?CM2, ?AV2 y ?CM3, ?AV3 respectivamente, con lo cual se representaría de esta manera {?CM1, ?CM2, ?CM3} $\in R_{camion\_X} \subseteq R_{camion}$, y {?AV1, ? AV2, ?AV3} $\in R_{avion\_Y} \subseteq R_{avion}$. A continuación en la tabla 2 se muestran todas las acciones que se forman desde O1, con las variables de recursos [35].

O1	
camión	*cargar(O1, X, ?CM1), cargar(O1, Y, ?CM1)* *mover(X, Y, ?CM1), mover(Y, X, ?CM1)* *descargar(O1, X, ?CM1), descargar(O1, Y, ?CM1)*
Avión	*embarcar(O1, X, ?AV1), embarcar(O1, Y, ?AV1)* *volar(X, Y, ?AV1), volar(Y, X, ?AV1), volar(Z, X, ?AV1)* *desembarcar(O1, X, ?AV1), desembarcar(O1, Y, ?AV1)*

Tabla 2. Acciones formadas para O1, con variables de recursos [35].

4.3.2.2 Segunda etapa: Expansión del grafo temporal

En esta etapa se expande un grafo temporal, moviéndose incrementalmente por instantes de tiempo *t*, generando en cada nivel acciones que consten de precondiciones y que no sean excluyentes en el mismo nivel, alternando acciones A_ty proposiciones P_t, culminando dicha expansión cuando las proposiciones de la situación final estén presentes y no sean excluyentes en un nivel de proposición [35].

En nuestro ejemplo si $t>D_{max}$, el algoritmo finaliza con fallo, como se muestra en la ilustración 27, ya que no existe ninguna clase de plan factible inferior a la duración o tiempo $D_{max.}$[35].

4.3.3.3 Tercera etapa: Obtención de un plan óptimo

En la ilustración 27 se observa la tercera etapa, que consta de tres procesos que se ejecutan de forma secuencial para llegar a un plan óptimo.

4.3.3.3.1 Plan abstracto.

El plan abstracto busca principalmente secuencias de acciones que satisfagan los objetivos del problema, tomando en cuenta que las acciones excluyentes no deben estar ejecutadas en paralelo, para cumplir con las siguientes propiedades [35]:

- El plan obtenido debe tener el mayor grado de paralelismo posible.
- El plan obtenido debe ser el más corto en lo que respecta a duración.

- El plan abstracto que se obtenga facilitará la escalabilidad del sistema.

4.3.3.3.2 Asignación de recursos

Llevada a cabo por un proceso de *scheduling*[24] que instancia todos los recursos necesarios a cada acción del plan abstracto [35].

4.3.3.3.3 Reparación del plan.

Se centra en conflictos de planificación, lo que conlleva a que dicho plan sea considerado como no factible. Dichos conflictos se obtienen secuenciando las acciones del plan y de ser el caso, este proceso agregará acciones necesarias para que el plan se lo considere como ejecutable, desplazándose a lo largo del grafo temporal generado en la segunda etapa. Sintetizando, los conflictos de planificación se resuelven agregando o añadiendo acciones que dejen disponibles los recursos para reutilizarlos [35].

4.4 Planificación Jerárquica

Si bien es cierto todos en el vivir diario necesitamos de cierto tipo de planificación para llevar un estilo de vida bueno, así por ejemplo, si queremos ir al cine: tenemos que acceder en primera instancia a una página de internet para revisar la cartelera, salimos de casa, nos dirigimos al cine, compramos un par de entradas e ingresamos a una sala. Se representan entonces los primeros pasos para resolver problemas de planificación [23].

Como se puede ver, este conjunto de pasos logró crear un proceso de planificación, pero no con la seguridad de resolver el plan de forma total, ya que cada una de las acciones en este caso serían: ingresar a internet, comprar las entradas, salir de casa [44]. Debido a la complejidad que suelen tener los problemas de planificación [11], se da lugar a la planificación jerárquica.

Se construye un plan con la información de las pautas iniciales acerca del dominio, que generalmente está formado por las acciones de alto nivel. Esto se hace con el fin de descomponerlas acciones en un grupo de varios niveles

[24]Es un componente funcional muy importante de los sistemas operativos multitarea y multiproceso, y es esencial en los sistemas operativos de tiempo real. Su función consiste en repartir el tiempo disponible de un microprocesador entre todos los procesos que están disponibles para su ejecución. Fuente:http://es.wikipedia.org/wiki/Planificador

de abstracción, que dan lugar a una jerarquía de acciones [2], con la ventaja de poder reutilizar sus planes que están ya laborados en otros problemas complejos
(Planificar acciones de bajo nivel desde cero podría resultar costoso)[14].
La planificación jerárquica se centra en la utilización de una versión abstracta del problema, haciendo más sencilla su elaboración, por lo que el problema que se consigue es refinado continuamente [46].
Para realizar una abstracción se deben tener en cuenta los siguientes niveles [11]:

- **A nivel de Literal.-** Utiliza el planificador *ABSTRIPS*, especifica los literales de nivel superior a través de la composición de los literales inferiores.
- **A nivel de acción.-** Sus objetivos son redes de tareas a resolver, muy distintos a los que se presenta en la planificación clásica, que básicamente busca alcanzar los objetivos mas no resolverlos.

La planificación jerárquica está formada por dos tipos de operadores o acciones [4]:

- **No Primitivos.-** Son las acciones que forman los niveles superiores que a su vez se descomponen en sub-tareas debidamente ordenadas, estas podrían ser acciones primitivas, pero para poder efectuar este operador el plan debe alcanzar los objetivos a partir de sus precondiciones, es decir el plan debe estar terminado.
- **Acciones Primitivas.-** Son las acciones de nivel inferior que ya no se pueden seguir descomponiendo.

Como se puede ver en la ilustración 29, se muestra un ejemplo de planificación jerárquica, la descripción del problema se hace de manera similar a lo que se hacía en la planificación clásica, es decir, con estados, operadores, precondiciones y efectos, que a su vez están guiados por tareas primitivas o no, primitivas junto con métodos que ayudan a descomponer tareas en sub-tareas [12].

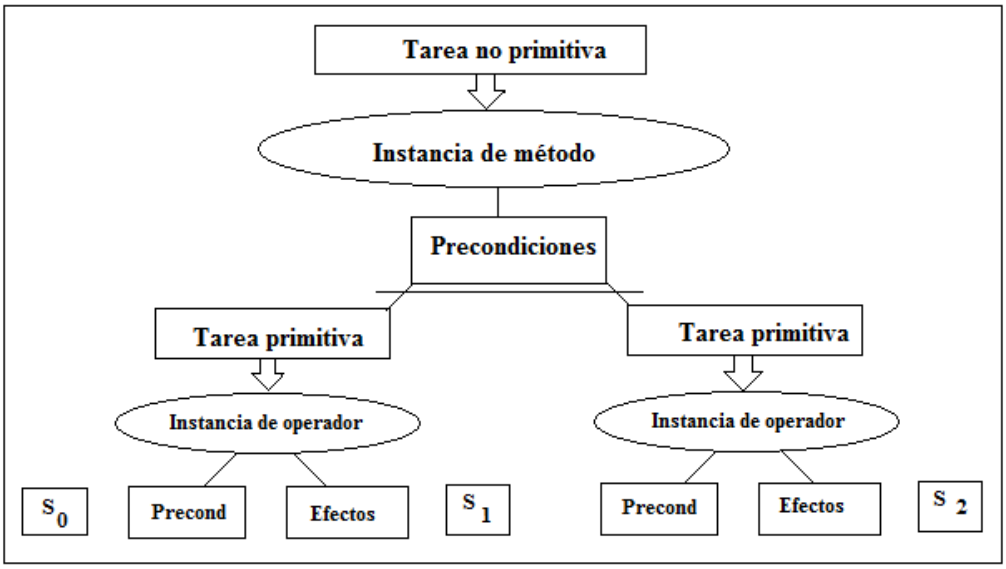


Ilustración 29. Planificación Jerárquica Simple – Descripción del problema [32].

- **Tarea.-** Se ve representa por la expresión *t* (*u*1,…,*un*), en donde , *t* simboliza la tarea y *u* es un término.
- **Método.-** Se especifica el nombre, tarea, precondiciones y las sub-tareas, dentro de una sola tupla:

m = (nombre (m), tarea (m), precond(m), subtareasm))

Ecuación 4: Descripción del problema - planificación jerárquica simple

- **Dominio de planificación.-** Se define los operados y métodos para resolver el problema.
- **Problema de planificación.-** Define el estado inicial, métodos, operadores y lista de tareas.
- **Solución.-** Se puede ejecutar planes de forma recursiva.

Las sub-tareas que son generadas por la descomposición de una tarea al cumplir con las restricciones y se conocen como HTN (*Heirarchical Task Network*).

4.4.1 Redes de Tareas Jerárquicas

Su funcionamiento es muy similar al del planificador *STRIPS* (apartado 2.2), ya que trabaja de manera recursiva e independiente de dominio, con cierto control sobre la búsqueda, con el objetivo de descomponer las acciones no

primitivas en abstracciones y lograr solo las acciones que puedan alcanzar los objetivos, es decir las primitivas [4].

Tienen como objetivo crear una secuencia de acciones que elaboren una tarea. Para describir su dominio necesita de un conjunto de operadores y métodos, utiliza estos métodos para descomponer las tareas en sub-tareas de manera recursiva, hasta que sus tareas sean primitivas, asi mismo, para las tareas que no sean primitivas es importante definir métodos aplicables para descomponer en aun más sub-tareas [2].

Ventajas:

- La información del dominio se detalla en acciones ya estructuradas de una manera genérica.
- Disminuye la complejidad de los problemas.
- Reduce la tarea del planificador, ya que solo tiene que modificar estas estructuras, ahorrando tiempo y costo en la resolución del problema [14].

Se presentan algunos planificadores con aplicaciones de HTN, cada uno tiene sus propias características y cuenta con su propio lenguaje para describir los dominios [37]:

- *SHOP*
- *O-PLAN*
- *SHOP2*
- *UMCP*
- *SIPE*
- *BRIDGE-BARON*

4.5 Planificación Por Incertidumbre

Si bien es cierto que resolver problemas de un entorno causa cierto tipo de incertidumbre, por el solo hecho de saber que se tiene que elaborar un proyecto, la causa se debe a que siempre se tiene inseguridad por los diferentes factores que podrían intervenir.

En la planificación clásica se definieron las características principales como deterministas, desarrolladas en un entorno observable para que sus metas

alcancen su objetivo, caso contrario, ocurre en la planificación por incertidumbre, por lo que se introducen los siguientes conceptos [38]:

- **No Determinista.-** Las acciones sobre los estados pueden cambiar, así también existen varias probabilidades de que los efectos al resolver el problema no sean los esperados, ya que en el mundo real y al momento de planificar nada es absolutamente predecible
- **Parcialmente Observable.-** Si bien es cierto resolver un problema requiere de la mayor información posible, resulta importante planificar en un entorno completamente observable, que permita al planificador disponer de esta información, ya que se busca obtener el conjunto de estados que forman la solución, cada estado a su vez se encuentra formado por distintas variables, que pueden o no ser observables.

  Al presentarse un problema parcialmente observable, ocasiona inconvenientes al momento de planificar, por lo que es mayor la desventaja en la planificación con incertidumbre, ya que el proceso de búsqueda podría llegar a ser infinito.
- **Metas Extendidas.-** Alcanzar la meta es el objetivo de solucionar un problema, por lo que es importante que esta meta se cumpla o garantizar que el estado en el que se encuentra sea seguro, cabe acotar que agregar metas extendidas a un problema puede llegar a ser difícil.

Ahora bien, la incertidumbre puede presentarse en los planificadores debido a que no siempre contarán con toda la información necesaria para solucionar problemas reales. A su vez, existe la posibilidad de que los objetivos no se puedan cumplir a exactitud o puedan ser modificados, ocasionando que las acciones simplemente no tengan éxito en su resultado, esta incertidumbre también puede ser causada por factores externos al problema.
A consecuencia de esto se han planteado los siguientes puntos [4][11]:

- A medida que se vaya ejecutando el plan, realizar una monitorización para poder desarrollar las modificaciones necesarias.
- Detectar y completar la información incompleta de un determinado entorno, de tal manera que las acciones realizadas por el planificador sean las adecuadas.
- Existen varios tipos de técnicas que ayudan en el proceso de planificación para reducir dicha incertidumbre, tales como: planificación conforme, contingente, reactiva, on-line. Por ello, utilizar estos planificadores, ayudará a prevenir ciertos fallos, elaborando estrategias para evitar en lo posible generar planes perjudiciales.
- Resolver los problemas mediante la toma de decisiones MDPs[25] es otra opción.

4.5.1 Proceso de Decisión Markov

El proceso de *Markov* es un proceso matemático, con técnicas que se han convertido en un estándar para la planificación basada en la toma de decisiones y aprendizaje por refuerzo. Dicho proceso asocia los estados y acciones en el espacio de estados por medio de una función de transición probabilística para poder crear un problema de decisión secuencial en un sistema que evoluciona en el tiempo [66], se presenta en un entorno observable con modelos no deterministas [39].

Propiedad de *Markov*.- Consiste básicamente en una acción que se genera sobre un estado, la misma que puede producir una distribución de probabilidades sobre sus posibles efectos [41].

- Para definir las acciones futuras necesita solo del estado actual, ya que no depende de acciones y estados anteriores.

[25] Consiste en aprender a decidir, ante una situación determinada, que acción es la más adecuada para lograr un objetivo: Fuente http://ocw.uc3m.es/ingenieria-informatica/t

El proceso de *Markov* se basa en los siguientes puntos para realizar una planificación [65]:

- Los planes se muestran como políticas de acción que maximicen la recompensa esperada en el tiempo.
- Entorno completamente observable del ambiente.
- El dominio del problema se crea como un modelo estocástico.
- Las metas se especifican por las funciones de utilidad /costo, para determinar la acción óptima en cada estado.
- El problema se ve como un problema de optimización.

Cada plan diseñado se presenta como una política de acción, en donde se selecciona un estado para definir una solución [40].
De igual manera, define una función de recompensa numérica para la acción del estado, con una función de utilidad, es decir, un estado se elabora a partir de bloques de nodo para hacer más fácil su construcción y determinar la utilidad de estados [38].
Para la resolución de este proceso se utiliza una tupla que define la planificación MDP, como un conjunto de estados y acciones, así como la probabilidad para pasar de una transacción a otra y el refuerzo que se recibe después de realizar un cambio [39].

$$< S, A, T, R >$$

Ecuación 5: Define la planificación MDP

En donde [40] [41]:

- $S \rightarrow$ Conjunto finito de estados
- $A \rightarrow$ Conjunto de posibles acciones
- $R \rightarrow$ Función de Recompensa que define un valor, tras producir una acción (a) en un estado (s).
- T (s; a; s′) → Modelo de Transición, (s; a; s′) la probabilidad de que se realice la transacción de pasar a un estado dado (s) el estado presente (s′) y la acción (a).

Ahora bien, el objetivo de la planificación es encontrar una política de MDP, que es una asociación $\pi\ S \rightarrow A,$ en donde, π representa la política en la que selecciona para cada estado S la acción que debe ser ejecutada para maximizar su valor.

Para evaluar la política se necesita evaluar la longitud de ejecución en los problemas mediante los siguientes criterios [38]:

- Problemas de horizonte finito.- El objetivo es maximizar la recompensa esperada total.
- Problemas de horizonte infinito.- En este caso cualquier política puede ser tanto buena o mala, al ser ejecutada durante un tiempo considerable,

4.5.1.1 Métodos de resolución

Los métodos principales para resolver el proceso de decisión *Markov* son por [38]:

- **Iteración de Política.-**Cuando se localiza una acción en un determinado estado y este estado tiene un valor mucho mejor que el de la acción actual, la política es mejora hasta encontrar una política optima
- **Iteración de Valor.-**Busca encontrar políticas óptimas de horizonte finito continuamente más largos hasta que convergen.

Adicional a la explicación de este proceso, detallamos algunas diferencias entre las diferentes técnicas de planificar, entre la planificación clásica y el proceso de decisión de *Markov* [38]:

MPDs	Planificación Clásica
Busca políticas	Busca acciones
Describe los estados a través de las instancias de las variantes relevantes de dominio	Sus estados estan definidos como un conjunto de cláusulas
Destacan la programaciándinámica	Su búsqueda la realizan en el espacio de estados y accciones
Cada acción que se produce en un estado, proporciona una recompensa, para definir las metas del problema.	Define las metas a través de la descripción de sus estados

Tabla 3. Diferencias entre la planificación clásica y el proceso de decisión *Markov* [38].

4.6. Paradigmas de planificación

Hoy en día existen una gran variedad de planificadores, debido a que se han sometido a evaluaciones que parten de tres intuiciones del ser humano, a continuación se expondrán los tres argumentos por los cuales los planificadores han avanzado en gran manera[2][48]:

- "*Divide y vencerás*", el cual claramente hace referencia a dividir un problema para su simplificación, en partes independientes llamadas sub-problemas, que a su vez se resuelven por separado. Los planes resultantes resuelven conflictos que existen entre ellos mediante una fusión entre los mismos.
- "*Abstracción jerárquica"*, hace referencia a la descomposición escalonada de un problema, encontrando una solución en el mayor nivel de abstracción.
- "*El conocimiento debe guiar la búsqueda*", en lo que se refiere a este punto, se enfoca en formalizar métodos que permitan utilizar diferentes tipos de conocimientos dependientes del dominio, durante la estructuración del plan.

A continuación se realizará un estudio breve de representación de estados:

Basado en estados.- Utiliza variantes de la representación *STRIPS (Standford Research Institute Problem Solver*), expresada por fórmulas llamadas metas, las cuales transforman el modelo inicial a otro modelo, probando la veracidad de dicha meta, además se basan en tres representaciones básicas, tal como se indica en [42].

- **Representación de estados:** dado por condiciones lógicas que parten de una descomposición, representando el estado como una secuencia de literales positivos conectados, siendo proporcionales o sin dependencias funcionales.
- **Representación de objetivos:** se representa a un objetivo como una secuencia de literales positivos y simples, por ende un objetivo *g*, es satisfecho siempre y cuando un estado proporcional *s* contenga todos sus elementos en *g*.
- **Representación de acciones:** Precondiciones que deben cumplirse antes de ejecutar las acciones, además de las consecuencias que conlleva la ejecución de la misma.

CAPITULO V

5.1 Definición de un problema de la vida real

Anteriormente hemos revisado algunos problemas que se pueden resolver mediante técnicas de planificación, ahora para poner en práctica lo estudiado hemos planteado un ejercicio que tiene diferentes etapas, desde la una básica o inicial, hasta la otra compleja.

Antes de establecer el problema, debemos tener una herramienta que pueda compilar el código fuente y así resolverlo, nosotros hemos optado por la herramienta PLTool, la cual cumple los requerimientos y estándares para la resolución de la aplicación (Anexo 1).

El ejercicio propuesto se trata sobre el traslado de objetos desde un punto inicial hasta un punto final, llamando punto a una localidad o ciudad, para nuestra primera etapa se contará con un conductor, un camión, dos ciudades que tienen conexión vial y 6 objetos o paquetes. En la ilustración 30 se puede observar el diagrama que representa mencionado problema en su etapa inicial.

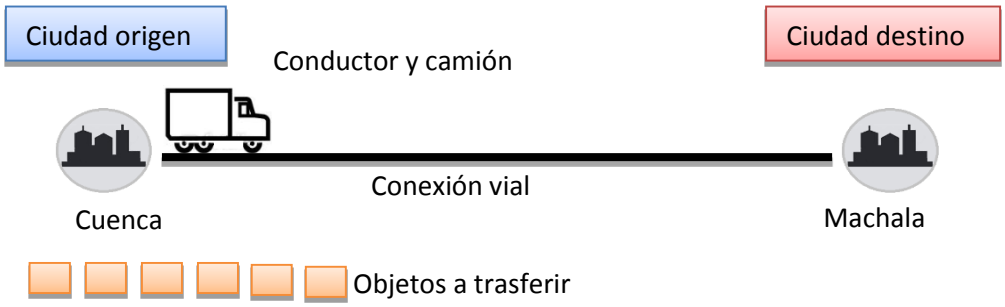


Ilustración 30. Problema de transporte de objetos en su primera etapa.

Antes de desarrollar el problema en cuestión en lenguaje PDDL, primero como recomendación se debe hacer una breve analogía del mismo y obtener resultado en lenguaje natural, para su posterior interpretación en el lenguaje PDDL.

Precondiciones:

Los objetos se cargan en los camiones para ser trasladados entre la ciudad origen y la ciudad destino.

- No existe límite en la capacidad de los camiones.
- Un camión necesita de un conductor para movilizarse.
- Debe existir conexión vial entre las ciudades origen y destino.

Acciones para **cargar camión**, parámetros (C1, Cm1, Obj1)

1. Que el camión Cm1 se encuentre en la ciudad origen C1.
2. Que el conductor Con1 se encuentre en el camión Cm1.
3. Que el objeto Ob1 se encuentre en la ciudad origen C1.
4. Que el objeto Obj1 no se encuentre dentro del camión Cm1.

Efectos

1. Que el objeto Obj1 se encuentra en el camión Cm1
2. Que objeto Ob1 no se encuentra en ciudad origen C1

Acciones para **transportar objeto**, parámetros (C1, C2, Ob1, Cm1)

1. Que el camión Cm1 no esté en ciudad destino C2.
2. Que exista conexión vial entre ciudad origen C1 y ciudad destino C2.
3. Que el objeto Obj1 se encuentre en camión Cm1.
4. Que el camión Cm1está en ciudad origen C1.

Efectos

1. Que el objeto Ob1 no se encuentra en ciudad origen C1.
2. Que el camión Cm1está en ciudad destino C2.

Acciones para **descargar objetos,** parámetros (Cm1, C2, Ob1)

1. Que el camión Cm1 se encuentre en ciudad destino C2.
2. Que el objeto Obj1 se encuentre en camión Cm1.

Efectos:

1. Que el objeto Obj1 se encuentre en ciudad C2.
2. Que el objeto Ob1 no está en camión cm1.

A continuación la representación en lenguaje PDDL:

- Definición del dominio representada por las ilustraciones 31 y 32:

```
(define (domain transporte-objetos)
  (:predicates (en-ciudad ?x  ?y)
               (en-camion ?x ?y)
               (conexion-vial ?x ?y)
               (objeto-en-ciudad ?x ?y)
               (objeto-en-camion ?x ?y)
               (conductor-en-ciudad ?x ?y)
               (camion-vacio ?x))
        (:action cargar-camion
               :parameters (?ciudadOrigen ?camion ?objeto
?conductor)
:precondition (and (en-ciudad ?camion ?ciudadOrigen) (en-camion
?camion ?conductor)  (objeto-en-ciudad ?objeto ?ciudadOrigen) (not
(objeto-en-camion ?objeto ?camion)))
:effect (and (not(objeto-en-ciudad ?objeto ?ciudadOrigen)) (objeto-
en-camion ?objeto ?camion))
        )
        (:action transportar-objeto
               :parameters(?ciudadOrigen ?ciudadDestino ?objeto
?camion)
:precondition (and  (not(en-ciudad ?camion ?ciudadDestino))
(conexion-vial ?ciudadOrigen ?ciudadDestino) (objeto-en-camion
?objeto ?camion))
:effect (and  (not(en-ciudad ?camion  ?ciudadOrigen)) (en-ciudad
?camion  ?ciudadDestino)))
```

Ilustración 31. Primera parte del código correspondiente a la definición del dominio del problema de transporte de objetos.

```
(:action descargar-objeto
                :parameters(?camion ?ciudadDestino ?objeto)
:precondition (and (en-ciudad ?camion ?ciudadDestino) (objeto-en-camion ?objeto ?camion) )
:effect (and (objeto-en-ciudad ?objeto ?ciudadDestino) (not(objeto-en-camion ?objeto ?camion)))
        )
)
```

Ilustración 32. Segunda parte del código correspondiente a la definición del dominio del problema de transporte de objetos.

- Definición del problema, representada por la ilustración 33.

```
(define (problem transporte)
 (:domain transporte-objetos)
(:objects objeto1  objeto2  objeto3  objeto4  objeto5 objeto6  cuenca machala  camion1 carlos)
(:init (en-ciudad camion1 cuenca) (objeto-en-ciudad objeto1 cuenca) (objeto-en-ciudad objeto2 cuenca) (objeto-en-ciudad objeto3 cuenca (objeto-en-ciudad objeto4 cuenca)  (objeto-en-ciudad objeto5 cuenca) (objeto-en-ciudad objeto6 cuenca)  (en-camion camion1 carlos) (conexion-vial cuenca machala) )
 (:goal (and (objeto-en-ciudad objeto1 machala) (objeto-en-ciudad objeto2 machala) (objeto-en-ciudad objeto3 machala) (objeto-en-ciudad objeto4 machala)  (objeto-en-ciudad objeto5 machala)  (objeto-en-ciudad objeto6 machala)) )
        )
```

Ilustración 33. Código correspondiente a la definición del dominio del problema de transporte de objetos.

En concordancia con la primera etapa de este problema, se ha compilado el dominio y la definición del problema por medio del planificador Metric-FF, que se fundamenta en realizar búsquedas heurísticas, ignorando los efectos negativos de las acciones [20], realizando resolución de planes relajados y con los resultados que arrojen estos planes se escoge el camino de menor distancia con referencia a la solución final [67].

En la ilustración se representan los resultados obtenidos, que son exitosos en esta primera y básica etapa inicial.

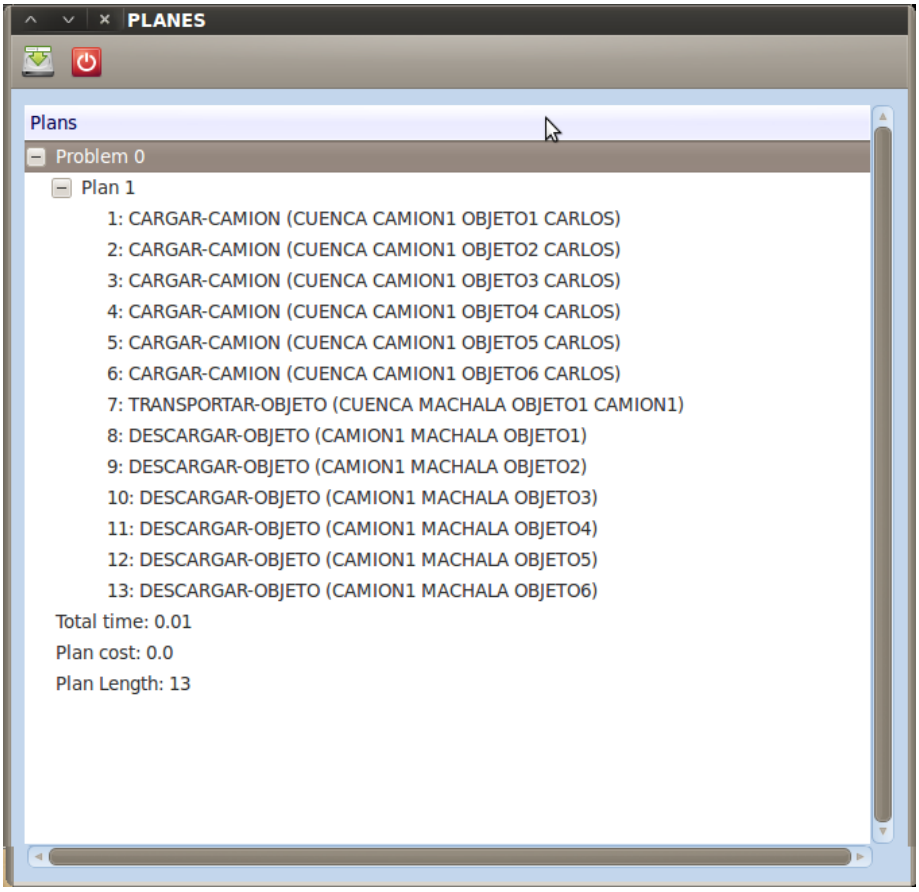


Ilustración 34. Resultado de compilación del problema de traslado de objetos

Una vez obtenidos los resultados en la primera etapa del problema, se procede a continuar y ejecutar una segunda, la cual tiene como objetivo trasladar 17 objetos que se encuentran en Cuenca, Loja, Machala, Ambato, hacia Quito, a su vez transportar 6 objetos que se encuentran en Machala hacia Cuenca. En la ilustración 35 se muestra el diagrama que representa dicha etapa del problema.

En esta etapa intervienen más factores que inciden en el desarrollo del problema, como más ciudades, más objetos y más camiones.

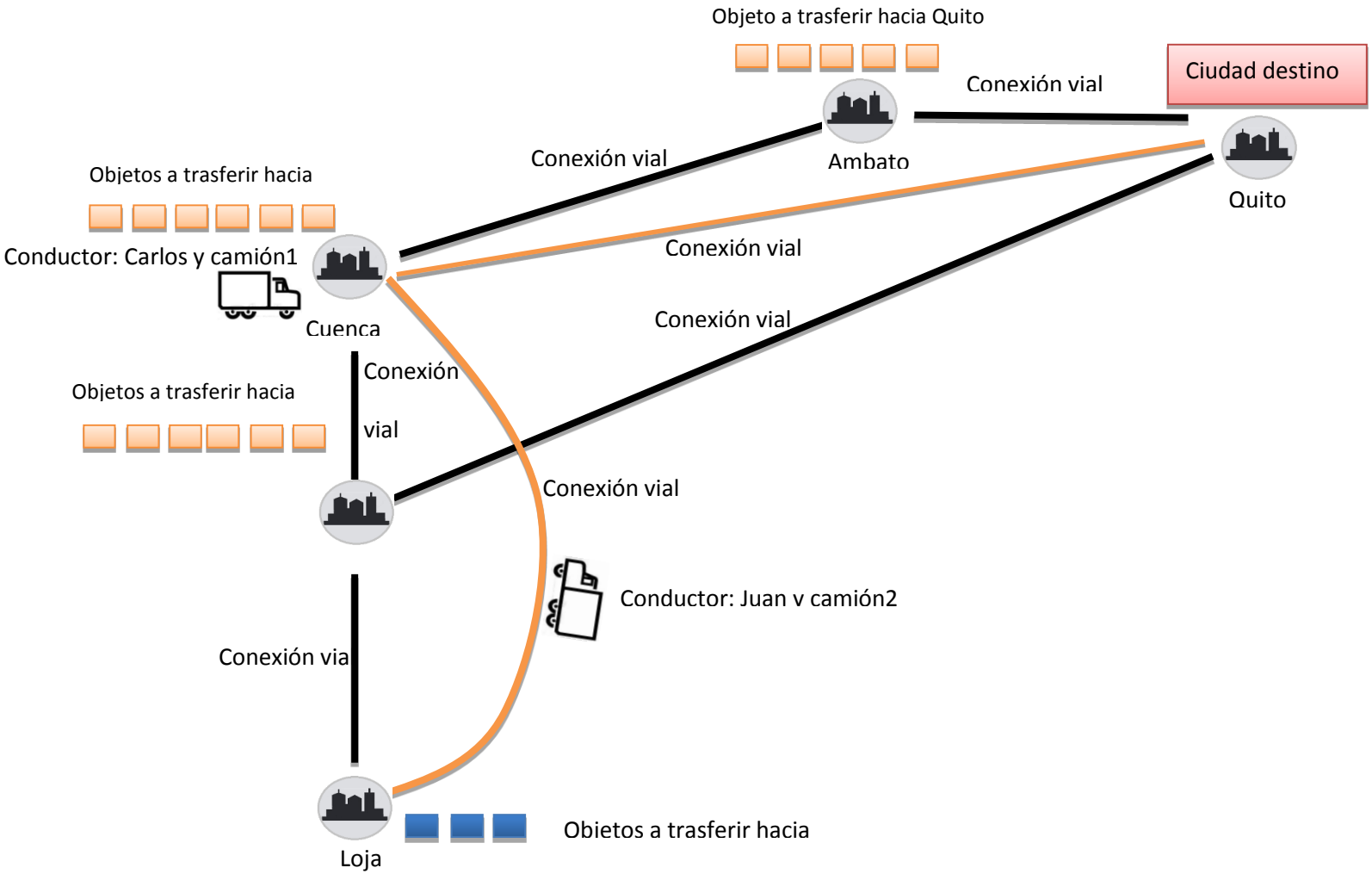


Ilustración 35. Representación del problema de traslado de objetos

En la ilustración 35, el problema presenta un grado de complejidad mayor al presentado en la etapa inicial, por esta razón en la definición del problema se debe especificar como arrancara el problema, la ilustración 36 muestra su correspondiente representación:

```
(define (problem transporte)
        (:domain transporte-objetos)
(:objects objeto1 objeto2 objeto3 objeto4 objeto5 objeto6 objeto7 objeto8
objeto9 objeto10 objeto11 objeto12  objeto13 objeto14 objeto15 objeto16
objeto17 objeto18 objeto19 objeto20 cuenca machala loja quito ambato
camion1 carlos camion2 juan)
(:init (en-ciudad camion1 cuenca) (en-ciudad camion2 loja) (objeto-en-
ciudad objeto1 cuenca) (objeto-en-ciudad objeto2 cuenca)  (objeto-en-
ciudad objeto3 cuenca) (objeto-en-ciudad objeto4 cuenca) (objeto-en-
ciudad objeto5 cuenca) (objeto-en-ciudad objeto6 cuenca)        (objeto-
en-ciudad objeto7 machala) (objeto-en-ciudad objeto8 machala) (objeto-
en-ciudad objeto9 machala)  (objeto-en-ciudad objeto10 machla) (objeto-
en-ciudad objeto11 machala) (objeto-en-ciudad objeto12 machala) (objeto-
en-ciudad objeto13 ambato) (objeto-en-ciudad objeto14 ambato) (objeto-
en-ciudad objeto15 ambato) (objeto-en-ciudad objeto16 ambato) (objeto-
en-ciudad objeto17 ambato) (objeto-en-ciudad objeto18 loja) (objeto-en-
ciudad objeto19 loja) (objeto-en-ciudad objeto20 loja) (en-camion camion1
carlos) (en-camion camion2 juan) (conexion-vial cuenca machala)
(conexion-vial cuenca loja) (conexion-vial cuenca ambato) (conexion-vial
cuenca quito) (conexion-vialmachala quito) (conexion-vialloja quito)
          (conexion-vialambato quito)
  )
(:goal (and (objeto-en-ciudad objeto1 quito) (objeto-en-ciudad objeto2
quito) (objeto-en-ciudad objeto3 quito) (objeto-en-ciudad objeto4 quito)
(objeto-en-ciudad objeto5 quito) (objeto-en-ciudad objeto6 quito) (objeto-
en-ciudad objeto7 quito) (objeto-en-ciudad objeto8 quito) (objeto-en-
ciudad objeto9 quito) (objeto-en-ciudad objeto10 quito) (objeto-en-ciudad
objeto11 quito) (objeto-en-ciudad objeto12 quito)  (objeto-en-ciudad
objeto13 quito) (objeto-en-ciudad objeto14 quito) (objeto-en-ciudad
objeto15 quito) (objeto-en-ciudad objeto16 quito) (objeto-en-ciudad
objeto17 quito) (objeto-en-ciudad objeto18 cuenca) (objeto-en-ciudad
objeto19 cuenca) (objeto-en-ciudad objeto20 cuenca))
  )
)
```

Ilustración 36. Código correspondiente a la definición del problema en su segunda etapa.

El problema representado por la ilustración 35, asciende un nivel mayor por motivo que intervienen factores como gasolina, distancias entre ciudades, cilindrajes de automotores y tiempos entre carga y descarga de objetos. El problema en su etapa final considera las siguientes condiciones, tomando en cuenta que los valores de dichas condiciones son valores a escala:

- Distancias:
  - Cuenca – Machala/ Machala-Cuenca : 3
  - Cuenca – Loja/ Loja - Cuenca : 4
  - Ambato – Quito/ Quito – Ambato : 2
  - Cuenca – Quito / Quito – Cuenca : 8
- Cilindraje de automotores
  - Primer camión con mayor consumo : 5
  - Segundo camión menor consumo: 3
- Nivel de gasolina
  - Primer camión: 70
  - Segundo camión: 50

En el siguiente punto se procederá a la resolución del problema de transporte de objetos, tomando en cuenta las condiciones citadas anteriormente.

5.2 Diseño de solución usando técnicas de Planning.

Una vez planteado el ejercicio en el cual se fundamenta este tema de tesis, se procede a explicar el planificador que se utilizó, junto con su respectivo algoritmo, además de su respectivo código fuente.

El planificador manejado para la resolución de este problema es LPG, que utiliza procedimientos eficientes para la resolución de problemas de satisfactibilidad y se basa en búsqueda local y grafos de planificación. LPG genera grafos a partir del grafo de planificación, aplicando algunas modificaciones para transformar un plan parcial en otro, estos planes son analizados por una función de evaluación parametrizada, la cual hace aplica diversas heurísticas [2] [6].

Además, LPG aplica el algoritmo A*, que encuentra el camino más corto entre dos puntos, utilizando la siguiente función de evaluación f(n) = g(n) + h(n), donde [9] [6]:

- h(n) = el valor heurístico del nodo a evaluar desde el actual, n, hasta el final.
- g(n) = el coste real del camino recorrido para llegar a dicho nodo, n.

Asimismo, usa combinaciones de búsquedas entre primero anchura y primero profundidad, en el instante en que h(n) tiende a profundidad, g(n) tiende a anchura, con lo cual existen cambio de nodos hasta encontrar la solución, si es que la existe [9].

A continuación el pseudocódigo del algoritmo A*, explicado en la ilustración 37:

```
ABIERTOS := [INICIAL]     //inicialización
CERRADOS := []
f'(INICIAL) := h'(INICIAL)
repetir
si ABIERTOS = [] entonces FALLO
si no                    // quedan nodos
extraer MEJORNODO de ABIERTOS con f' mínima
// cola de prioridad
mover MEJORNODO de ABIERTOS a CERRADOS
si MEJORNODO contiene estado_objetivo entonces
SOLUCION_ENCONTRADA := TRUE
si no
generar SUCESORES de MEJORNODO
para cada SESOR hacer TRATAR_SUCESOR
hasta SOLUCION_ENCONTRADA o FALLO
```

Ilustración 37. Explica el pseudocódigo del algoritmo A*.

En resumen este algoritmo es muy útil al calcular el camino óptimo entre dos puntos, basándose en costes, ósea si sabemos cuál es el valor de todo el

movimiento del camino y sabemos el valor del camino, podemos estimar su eficiencia [68].

Se ha realizado un breve análisis del planificador y el algoritmo utilizado en nuestro problema, a continuación se representa el ejercicio en su etapa final, con su respectivo código fuente desarrollado en PDDL:

- En primera instancia se debe definir el dominio del problema donde estableceremos todas las acciones del mismo, junto con sus respectivos efectos, en la siguientes ilustraciones 38 y 39, se expone el código fuente:

```
(define (domain transporte-objetos)
(:requirements :strips :typing :fluents :durative-actions)
(:types camion caja conductor lugar - object  ciudad - lugar)
  (:predicates (en-ciudad ?x -camion  ?y - ciudad)
               (en-camion ?x - camion ?y - conductor)

(conexion-vial ?x - ciudad ?y - ciudad)
               (objeto-en-ciudad ?x - caja ?y - ciudad)
               (objeto-en-camion ?x - caja ?y - camion))
(:functions (nivel_gasolina ?x - camion) (total_gasolina)
(consumo_gasolina ?x - camion) (distancia ?x - ciudad ?y - ciudad)
(porcentaje_consumo ?x - camion))
(:durative-action cargar-camion  :parameters (?ciudadOrigen -
ciudad ?camion - camion ?objeto - caja ?conductor - conductor)
:duration (= ?duration 30)
:condition (and (over all (en-ciudad ?camion ?ciudadOrigen)) (over
all (en-camion ?camion ?conductor))  (at start (objeto-en-ciudad
?objeto ?ciudadOrigen)) (at start (not (objeto-en-camion ?objeto
?camion))))
:effect (and (at end (not(objeto-en-ciudad ?objeto ?ciudadOrigen)))
(at start (objeto-en-camion ?objeto ?camion)))
)
(:durative-action transportar-objeto
:parameters(?ciudadOrigen - ciudad ?ciudadDestino - ciudad
?objeto - caja ?camion - camion)
:duration (= ?duration (* 2 (distancia ?ciudadOrigen
?ciudadDestino)))
:condition (and (at start (>= (nivel_gasolina ?camion) (* (distancia
?ciudadOrigen ?ciudadDestino) (porcentaje_consumo ?camion))))
(at start (not (en-ciudad ?camion ?ciudadDestino))) (over all
(conexion-vial ?ciudadOrigen ?ciudadDestino)) (at start (en-ciudad
?camion ?ciudadOrigen)) (over all (objeto-en-camion ?objeto
?camion)))
:effect (and  (at end (not(en-ciudad ?camion  ?ciudadOrigen))) (at
end (decrease (nivel_gasolina ?camion) (* (distancia ?ciudadOrigen
?ciudadDestino) (porcentaje_consumo ?camion))))  (at end (en-
ciudad ?camion  ?ciudadDestino)) (at end (increase (total_gasolina)
1)))
```

Ilustración 38. Corresponde a la primera parte del código fuente, en donde se define el dominio del problema.

```
 (:durative-action descargar-objeto
:parameters(?camion - camion ?ciudadDestino - ciudad ?objeto - caja)
:duration (= ?duration 20)
:condition (and (over all (en-ciudad ?camion ?ciudadDestino)) (at start
(objeto-en-camion ?objeto ?camion)) )
:effect (and (at end (objeto-en-ciudad ?objeto ?ciudadDestino)) (at end
(not(objeto-en-camion ?objeto ?camion))))
          )
)
```

Ilustración 39. Corresponde a la segunda parte del código fuente, en donde se define el dominio del problema.

- A continuación se procede a definir el problema, con sus condiciones iniciales y sus respectivos objetivos, en las ilustraciones 40 y 41, se muestra el código fuente del mismo:

```
(define (problem transporte)
  (:domain transporte-objetos)
  (:objects objeto1 - caja objeto2 - caja objeto3 - caja objeto4 - caja objeto5 -
caja objeto6 - caja objeto7 - caja objeto8 - caja objeto9 - caja objeto10 - caja
objeto11 - caja objeto12 - caja objeto13 - caja objeto14 - caja objeto15 - caja
objeto16 - caja objeto17 - caja objeto18 - caja objeto19 - caja objeto20 - caja
cuenca - ciudad machala - ciudad loja - ciudad   ambato - ciudad  quito -
ciudad camion1 - camion carlos -
          conductor camion2 - camion juan - conductor)
(:init (en-ciudad camion1 cuenca)(en-ciudad camion2 loja) (objeto-en-ciudad
objeto1 cuenca)(objeto-en-ciudad objeto2 cuenca)(objeto-en-ciudad objeto3
cuenca)(objeto-en-ciudad objeto4 cuenca)(objeto-en-ciudad objeto5
cuenca)(objeto-en-ciudad objeto6 cuenca)(objeto-en-ciudad objeto7
machala) (objeto-en-ciudad objeto8 machala)(objeto-en-ciudad objeto9
machala) (objeto-en-ciudad objeto10 machala)(objeto-en-ciudad objeto11
machala)(objeto-en-ciudad objeto12 machala)(objeto-en-ciudad objeto13
ambato) (objeto-en-ciudad objeto14 ambato) (objeto-en-ciudad objeto15
ambato) (objeto-en-ciudad objeto16 ambato) (objeto-en-ciudad objeto17
ambato)(objeto-en-ciudad objeto18 loja) (objeto-en-ciudad objeto19 loja)
(objeto-en-ciudad objeto20 loja) (en-camion camion1 carlos) (en-camion
camion2 juan)(conexion-vial cuenca machala)(conexion-vial machala
cuenca)(conexion-vial cuenca loja)(conexion-vial loja cuenca) (conexion-vial
machala ambato)(conexion-vial ambato machala) (conexion-vial ambato
quito)(conexion-vial quito ambato) (conexion-vial cuenca quito)(conexion-
vial quito cuenca)(= (nivel_gasolina camion1) 70)(= (nivel_gasolina camion2)
50)(=  (consumo_gasolina camion1) 5) (= (consumo_gasolina camion2) 3) (=
(total_gasolina) 0)(= (distancia cuenca machala) 3) (= (distancia cuenca loja)
4) (=
```

Ilustración 40. Corresponde al código fuente que define el problema planteado.

```
(distancia machala cuenca) 3) (= (distancia loja cuenca) 4)  (= (distancia
machala ambato) 3)  (= (distancia  ambato machala) 3) (= (distancia  ambato
quito) 2)  (= (distancia  quito ambato) 2) (= (distancia  cuenca quito) 8)  (=
(distancia  quito cuenca) 8) (= (porcentaje_consumo camion1) 2) (=
(porcentaje_consumo camion2) 4) )
(:goal (and (objeto-en-ciudad objeto1 quito) (objeto-en-ciudad objeto2 quito)
(objeto-en-ciudad objeto3 quito) (objeto-en-ciudad objeto4 quito) (objeto-
en-ciudad objeto5 quito) (objeto-en-ciudad objeto6 quito) (objeto-en-ciudad
objeto7 quito) (objeto-en-ciudad objeto8 quito) (objeto-en-ciudad objeto9
quito) (objeto-en-ciudad objeto10 quito) (objeto-en-ciudad objeto11 quito)
(objeto-en-ciudad objeto12 quito) (objeto-en-ciudad objeto13 quito) (objeto-
en-ciudad objeto14 quito) (objeto-en-ciudad objeto15 quito) (objeto-en-
ciudad objeto16 quito) (objeto-en-ciudad objeto17 quito) (objeto-en-
ciudad objeto18 cuenca) (objeto-en-ciudad objeto19 cuenca) (objeto-en-
ciudad objeto20 cuenca)))
  (:metric minimize (total-time)))
```

Ilustración 41. Corresponde a la segunda parte del código fuente que define el problema planteado.

Al compilar los dos archivos con su correspondiente código fuente, el analizador generó una serie de planes resultantes, como se puede observar en la ilustración 42. Se obtuvieron un total de dos planes, con sus correspondientes costes, tiempos iteraciones.

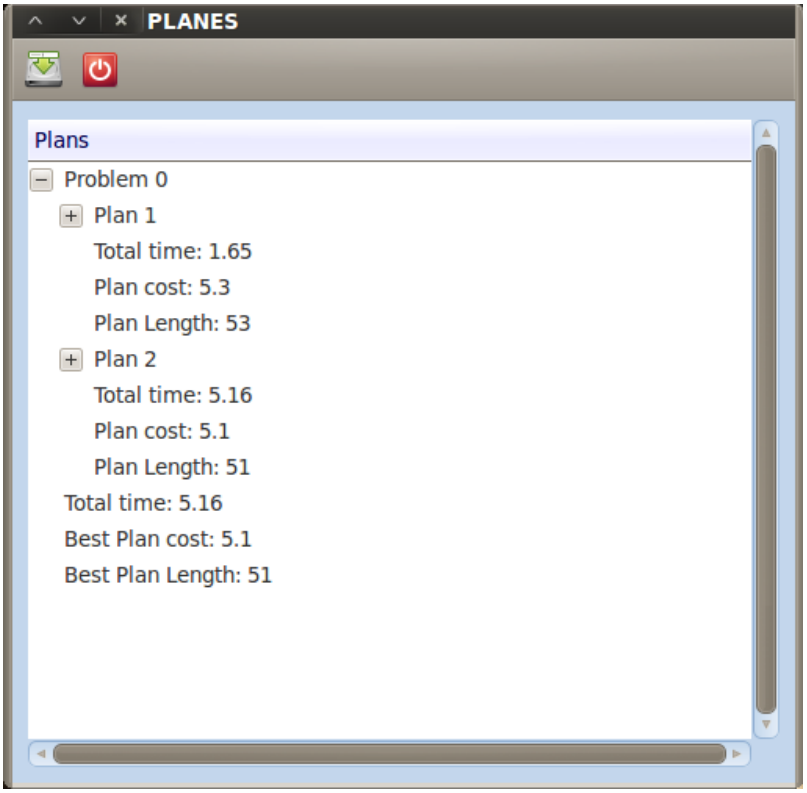


Ilustración 42. Resultados del problema en su etapa final.

Adicionalmente, se ha realizado una pequeña aplicación en donde se simulan los resultados que genera el analizador, ya que dicha herramienta genera archivos XML, con sus respectivos operadores, atributos y resultados. Esta aplicación se realizó en el lenguaje de programación Java, y tiene como finalidad llevar a cabo una animación gráfica del problema, examinando el archivo XML y ejecutando cada una de las instrucciones que se encuentran en el mismo (Anexo 2).

5.3 Diseño de la solución usando técnicas clásicas

En punto anterior se pudo observar cómo se aplicaron técnicas de planificación, ahora se pretende aplicar otras técnicas, las cuales busquen resolver y dar solución al mismo problema que se planteó para el desarrollo de esta tesis.

A pesar de que el problema planteado se lo pudo resolver de manera satisfactoria, pretendemos aplicar otras técnicas tales como CSP (Problemas de Satisfacción de Restricciones) y algoritmos genéticos. En el siguiente texto se explicarán estas dos técnicas y analizaremos que resultados obtenemos de las mismas.

- **CSP (Problemas de satisfacción de restricciones)**.- Es una metodología utilizada para la resolución de problemas de optimización, buscando la satisfabilidad del mismo, obteniendo una o varias soluciones, ya sea una solución óptima, en base a una función objetivo previamente definida en términos de algunas o todas las variables [71].

  CSP tiene las siguientes características:

  - El problema es constituido por medio de un conjunto de variables.
  - Cada variable contiene un dominio de valores.
  - Existen restricciones de consistencia entre las variables binarias o n-arias.
  - Una solución es una asignación de valores a esas variables.

  Los problemas de satisfacción de restricciones pueden ser representados por la tupla {X, D, C}, donde [16]:

  - *X* es un conjunto finito de variables
  - *D* es una función que mapea cada variable $X_i \in X$ al dominio $D(X_i)$, que es un conjunto finito de posibles valores que se puede asignar a X_i.
  - *C* es el conjunto de restricciones, descrito por la relación de algunas variables.

  Para el desarrollo de un problema de satisfacción de restricciones se debe tener en cuenta los siguientes puntos [16]:

- **Solución**.- Una asignación es un conjunto de pares que representa la asignación simultanea de los valores v_i, para cada X_i, para las n variables del problema.
- **Insatisfacción de restricciones.-** Suceden cuando alguna asignación utiliza algunas de las combinaciones descritas en el conjunto de restricciones C.
- **Parámetros del problema.-** Existen parámetros que permiten clasificar los CSP ya que varían en cuanto a tamaño y nivel de dificultad, por esa razón los parámetros que definen el problema son:
    - n: número de variables del problema.
    - m: Tamaño del dominio.
    - $p1 \in (0,1)$: Medida de conectividad, definiendo la cantidad de restricciones.
    - $p2 \in (0,1)$: Medida de incompatibilidad, definiendo los valores incompatibles para cada restricción.

- **Algoritmos genéticos (AGs).-** Utilizados para la resolución de problemas de búsquedas y optimización, imitando el proceso de evolución propuesto por Darwin (1859), el cual hace mención a la evolución natural de las poblaciones y la supervivencia de los más fuertes. Los AGs son capaces de construir soluciones para el mundo real, trabajando con una población de individuos, los cuales representan soluciones factibles a un problema dado [69].

  Los AGs presentan algunas características generales tales como [70]:

    - No necesitan conocer el problema en su totalidad para poder resolverlo.
    - Es una búsqueda robusta, utilizando operadores probabilísticos a diferencia de otras técnicas.

- No trabajan de manera secuencial, es decir, trabajan de manera simultánea con varias soluciones.
- Cuando son utilizados para maximizar una función objetivo, es decir, problemas de optimización, son los que menos afectados resultan por falsas soluciones en comparación con otras técnicas.

A continuación en la ilustración 43, se muestra el pseudocódigo del algoritmo genético simple

```
BEGIN /* Algoritmo Genético Simple */
Generar una población inicial.
Computar la función de evaluación de cada individuo.
WHILE NOT Terminado DO
BEGIN /* Producir nueva generación */
FOR Tamaño población/2 DO
BEGIN /*Ciclo Reproductivo */
        Seleccionar dos individuos de la anterior
        generación,
        para el cruce (probabilidad de selección
        proporcional
        a la función de evaluación del individuo).
        Cruzar con cierta probabilidad los dos
        individuos obteniendo dos descendientes.
        Mutar los dos descendientes con cierta
        probabilidad.
        Computar la función de evaluación de los dos
        descendientes mutados.
        Insertar los dos descendientes mutados en la
        nueva generación.
END
```

Ilustración 43. Pseudocódigo del Algoritmo genético simple.

Ahora definiremos algunos conceptos básicos biológicos como son [70]:

- **Cromosoma**.- Es la cadena de ADN y proteínas.
- **Gen**.- Los cuales forman los cromosomas y determina rasgos de un individuo.

- **Alelos**.- Posibilidades de escoger o elegir un rasgo.
- **Locus**.- Posición en la que se encuentra el gen, dentro del cromosoma.
- **Genoma**.- Conjunto de material genético.
- **Genotipo**.-Conjunto de genes.
- **Fenotipo**.-Conjunto de características finales físicas y mentales del individuo.

Una vez establecida una breve descripción algunos conceptos biológicos, pasaremos a la descripción de los mismos conceptos biológicos en términos de AGs [70]:

- **Cromosoma**.- Cadena que codifica cada solución.
- **Gen**.- Pude ser representado por un bit o bloques cortos de bits adyacentes que codifican un elemento particular del candidato a solución.
- **Alelos**.- Valor de una característica.
- **Locus**.- Posición en la cadena que representa el cromosoma.
- **Genotipo**.- Es la configuración de bits del cromosoma del individuo.
- **Fenotipo**.-Estructura que está sometida al problema,
- **Generación**.-Representa el ciclo.

Selección.- Es el proceso en donde se escogen los individuos de la población mejor adaptados, estos individuos servirán para la reproducción o para pasar a la siguiente generación, es decir, se eligen a los progenitores de la siguiente generación, estos progenitores se cruzaran genéticamente y generan una nueva descendencia [36].

Entre los operadores genéticos más frecuentes se encuentran la mutación, el crossover y la reproducción. Para nuestro estudio y desarrollo del problema nos centraremos en la mutación.

Cruce.- Es el proceso en donde se mezclan los individuos que pasaron por la fase de selección, combinándose los genes de los dos padres entre sí, generando así nuevos hijos. Para la resolución del problema se utilizó el cruce basado en un punto, que no es más que la mezcla de los dos padres por medio de un punto de

corte, intercambiando secciones, es decir, que todos los genes desde el punto de corte hacia la izquierda del padre1 forman parte del hijo1, y los restantes o los que quedan hacia la derecha del punto de corte forman parte del hijo2, dándose el mismo caso para el padre2. En la ilustración 44, se puede observar un ejemplo de cruce.

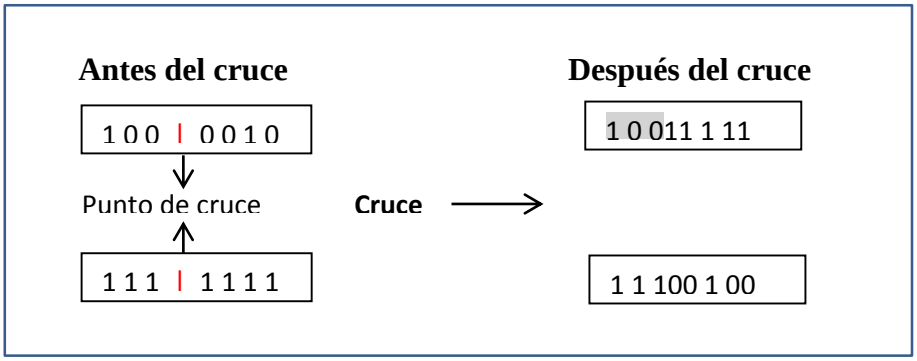


Ilustración 44. Ejemplo de cruce de genes [36].

Mutación, que se aplica a cada hijo por separado, no es más que la alteración aleatoria de cada gen, que es el componente del cromosoma [7], claro está, dicha alteración está basada en una probabilidad pequeña, la cual indica que si el número generado aleatoriamente está por debajo de esta probabilidad se cambiara el bit (de 1 a 0, o de 0 a 1), caso contrario no existirá cambio y lo dejara como estaba, en la ilustración 45 se puede observar el proceso de mutación [8].

Un aspecto a tomar en cuenta es que dependiendo del número de individuos y del número de bits por individuo, las mutaciones pueden resultar muy extrañas o raras en una sola generación, por ese motivo no es bueno abusar de la mutación.

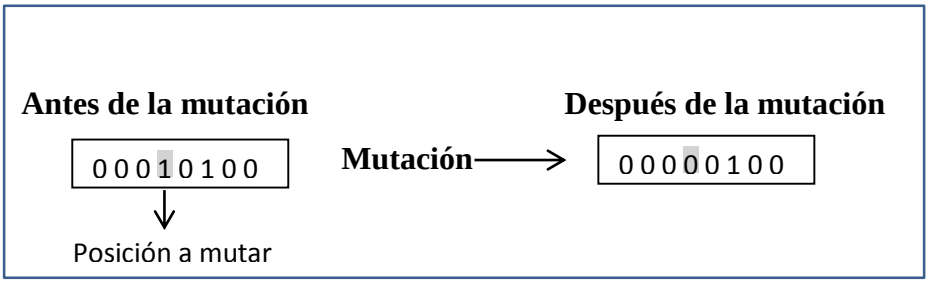


Ilustración 45. Ejemplo de mutación [36].

Función objetivo (*fitness*).- Es la que permite valorar las aptitudes de los individuos, es decir establece que tan bueno es el conjunto de genes para conseguir un objetivo, ésta función siempre toma valores positivos y se construye o establece según el problema [52], [36].

Se ha realizado un breve estudio de dos técnicas de planificación como son los CSP y algoritmos genéticos, se ha podido analizar que ninguna de las dos técnicas satisfacen las necesidades del problema planteado, ya que por un lado las dos metodologías buscan resolver problemas de búsquedas y optimización, no toman en cuenta factores como el tiempo o duración de cada paso, coste del mismo, consumo de gasolina, ni cilindraje. Al referirnos a algoritmos genéticos se podría encontrar alguna solución, pero realizando un algoritmo combinado, es decir un algoritmo híbrido, que tiene un grado de dificultad muy complejo ya que combina técnicas de planificación y algoritmos genéticos para encontrar una solución.

5.4 Comparación de resultados

Una vez realizado el análisis del problema por medio de varias técnicas de planificación, se ha podido observar que las técnicas clásicas como CSP y algoritmos genéticos no generaron resultados satisfactorios, ya que si bien son metodologías que buscan optimizar los problemas de búsqueda, no se pudieron aplicar a nuestro problema, por factores como el nivel de gasolina, cilindraje y

distancias de rutas y por ende, no pudieron ser representados en conjunto por las técnicas mencionadas anteriormente. Por otro lado, las técnicas de planificación que aplicamos resolvieron el problema, dando como resultado el analizador dos planes en los cuales se representa el tiempo, iteraciones y costo del mismo, de los mismos el primer plan satisface en totalidad el problema que hemos establecido, ya que sigue una secuencia lógica de pasos, minimizando tiempo y costos, en las ilustraciones 46, 47, 48, 49, 50 y 51 se pude observar los dos mejores planes, basados en mejor tiempo y en menor número de iteraciones.

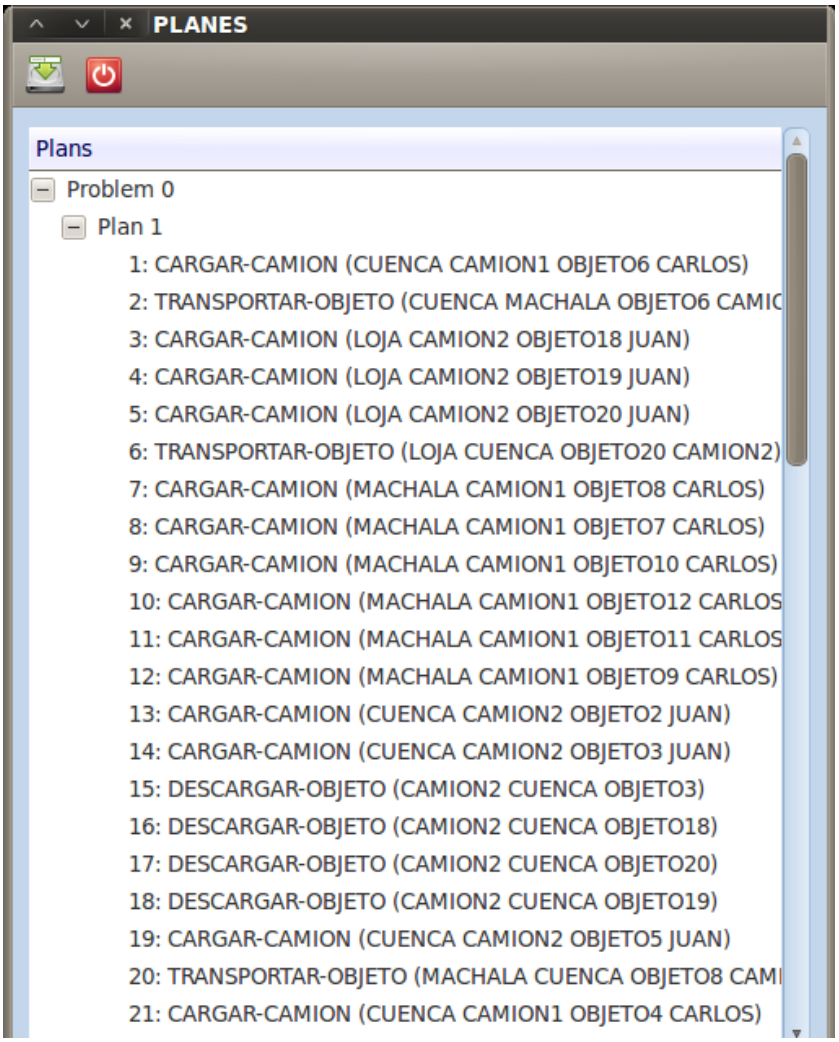


Ilustración 46. Primera parte del plan éxito con menor tiempo, correspondiente al plan uno, resultante del análisis del problema de traslado de objetos.

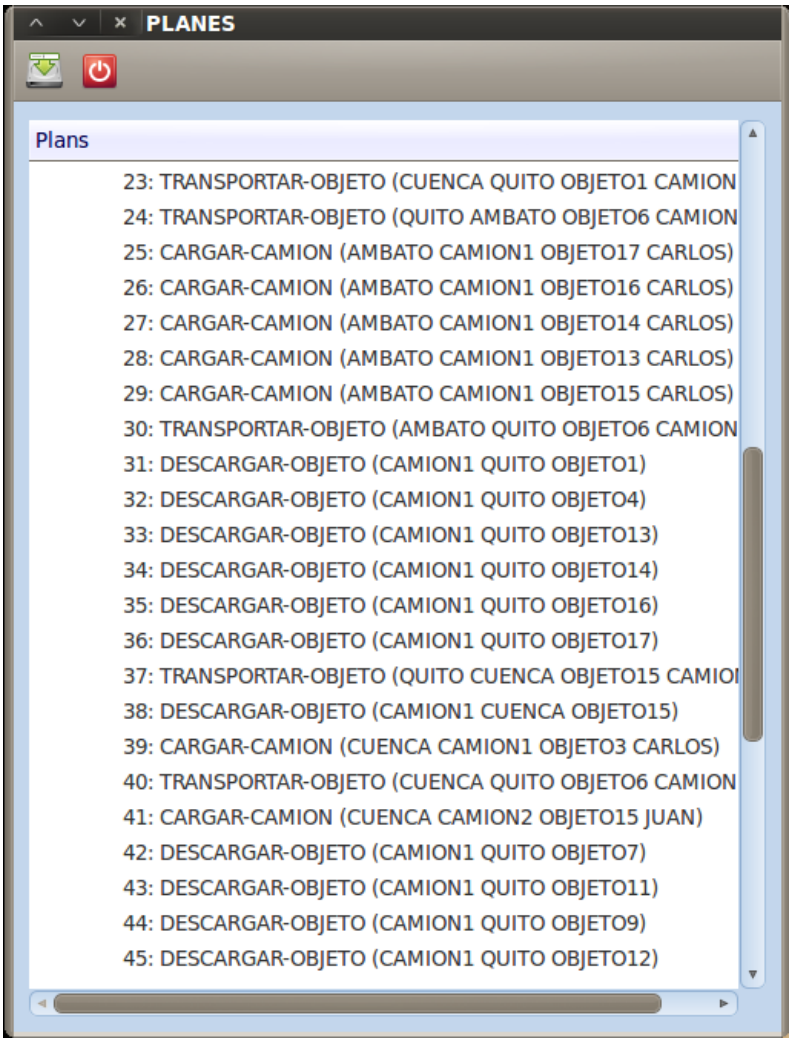


Ilustración 47. Segunda parte del plan éxito con menor tiempo, correspondiente al plan uno, resultante del análisis del problema de traslado de objetos.

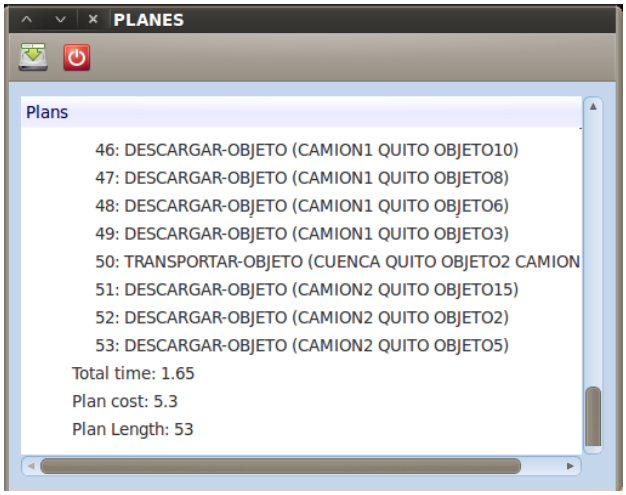


Ilustración 48. Tercera parte del plan éxito con menor tiempo, correspondiente al plan uno, resultante del análisis del problema de traslado de objetos.

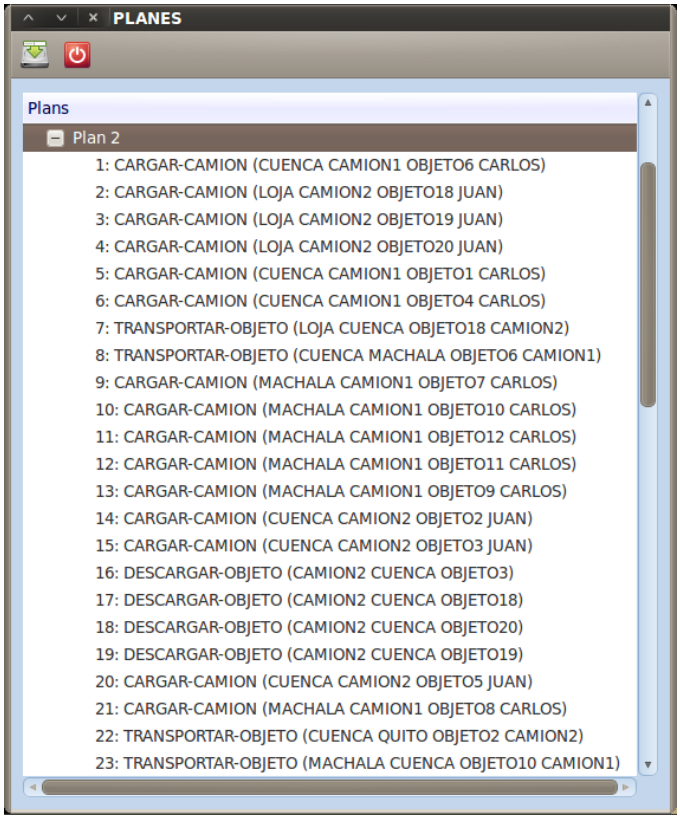


Ilustración 49. Primera parte del plan éxito con menor número de iteraciones, correspondiente al plan dos, resultante del análisis del problema de traslado de objetos.

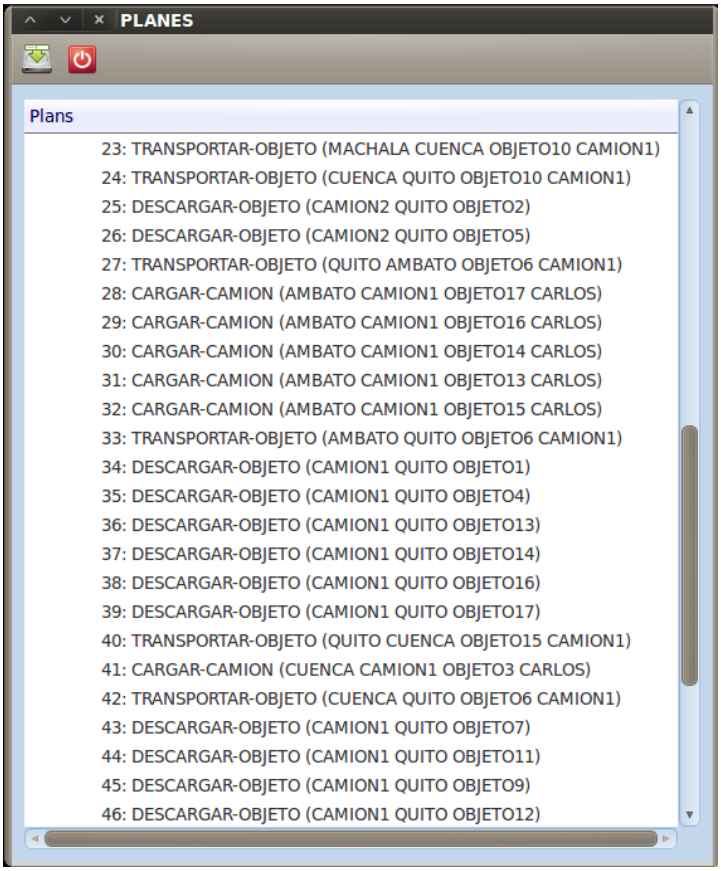


Ilustración 50. Segunda parte del plan éxito con menor número de iteraciones, correspondiente al plan dos, resultante del análisis del problema de traslado de objetos

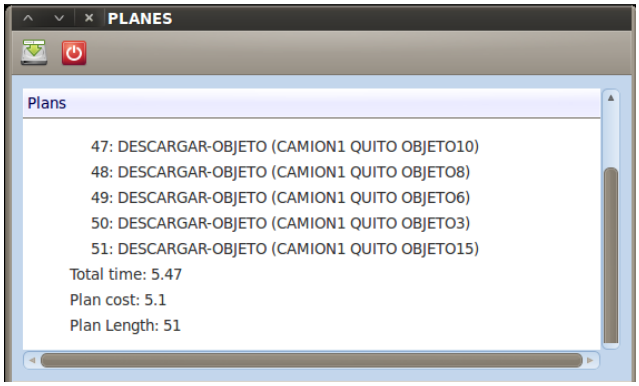


Ilustración 51. Tercera parte del plan éxito con menor número de iteraciones, correspondiente al plan dos, resultante del análisis del problema de traslado de objetos.

Como se puede observar en las ilustraciones 49, 50 y 51, correspondientes a los resultados del plan número dos, los resultados generados en cuanto al número de iteraciones y costos son mucho más óptimos que el plan número uno, aunque en tiempos el último plan mencionado supera en gran escala los resultados del segundo plan, resolviendo el problema tres veces menos el valor del tiempo del plan dos, el valor exacto del tiempo del plan uno es de 1.65, mientras que el plan dos presenta un tiempo de 5.47.

En cuanto a algoritmos genéticos se ha mencionado que no satisfacen las necesidades totales del problema planteado, se ha realizado una simulación de cómo encontrar una ruta valida entre seis ciudades, en donde las ilustraciones 1 y 2 muestran los resultados obtenidos:

Ilustración 52. Gráfica que representa el número de iteraciones en nuevas poblaciones mutadas, en intervalos de tiempo de milisegundos.

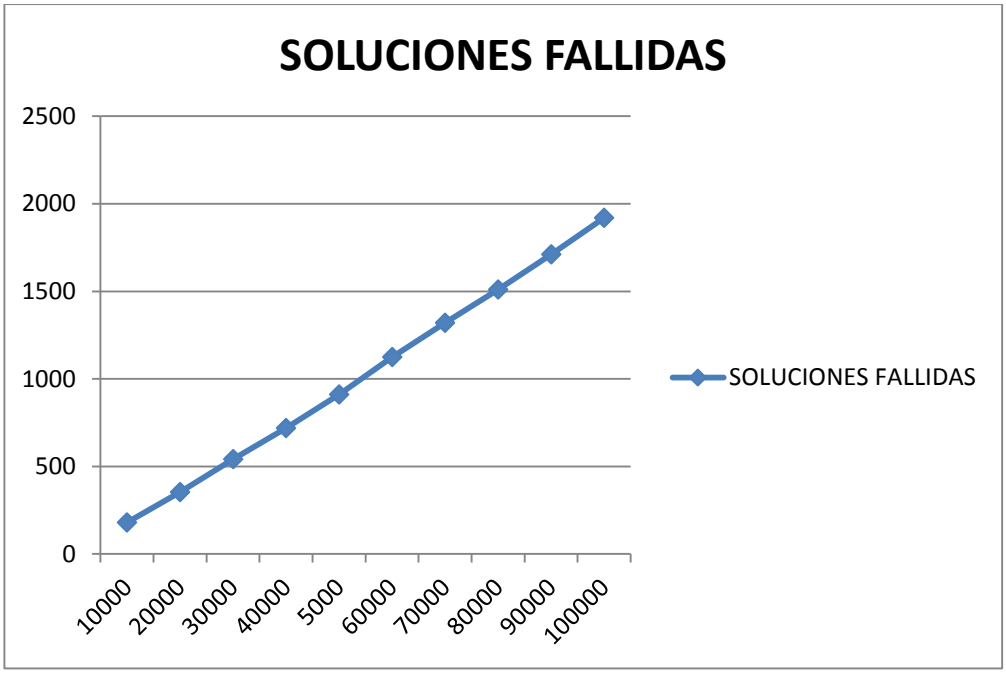


Ilustración 53. Gráfica que representa el número de soluciones fallidas para encontrar una ruta óptima, en intervalos de tiempo de milisegundos.

Como se puede observar en las ilustraciones 52 y 53, los algoritmos genéticos si bien son una metodología robusta y eficiente en muchos campos, no cumple con los requerimientos que el problema solicitados, ya que por un lado se puede modelar la conexión entre ciudades, se podría encontrar una solución aplicando y probando diferentes funciones de *fitness*, hasta encontrar una que encuentre una solución, pero la maximización de combustible, distancia entre ciudades, y cilindraje de los automotores requieren un modelado más complejo y completo. Como anteriormente se ha mencionado se pueden representar los factores previamente mencionados por medio de técnicas combinadas, es decir, combinación entre técnicas clásicas y algoritmos genéticos, lo que da como resultado los algoritmos híbridos, los cuales tienen un grado alto de complejidad en su desarrollo.

5.5 Análisis y discusión

Durante la ejecución y comparación de resultados del problema de transporte de objetos, se pudo observar que los dos planes resultantes satisfacían las necesidades del mismo, de los cuales el primero es el que tiene mejor tiempo, mientras que el plan número dos presenta mayor tiempo en su ejecución, aunque genera mejores resultados en cuanto a número de iteraciones y costos. Realizando una breve síntesis los resultados obtenidos por las técnicas de planificación, resultaron satisfactorios para la resolución y modelaje del problema, produciendo más de una solución. Por lo antes mencionado, se puede decir que las técnicas de planificación son metodologías muy útiles, las cuales resuelven problemas en campos como la robótica, medicina, telecomunicaciones, industria, transporte etc.

En lo que se refiere a CSP y algoritmos genéticos, son técnicas muy robustas y eficientes en el campo de la optimización y búsqueda de soluciones, por un lado, en CSP no se pudo modelar factores claves como maximización de combustible, distancia entre rutas y cilindraje de los automotores, los cuales inciden en el desarrollo del problema, y por otro lado, con algoritmos genéticos se pudo modelar parte del problema, teniendo éxito en encontrar conexiones entre ciudades, aunque resulta muy complejo la representación de los factores citados anteriormente, ya que se requiere de un estudio minucioso y realización de representaciones de técnicas o metodologías combinadas para encontrar una o varias soluciones óptimas, las cuales satisfagan las necesidades del problema. Sería muy interesante realizar el estudio de este problema de tesis por medio de algoritmos híbridos y analizar los resultados que genere el mismo, realizando una comparación a mayor detalle entre las técnicas de planificación y algoritmos híbridos.

CONCLUSIONES

Al concluir la presente tesis, hemos podido observar que el modelado del problema central de este trabajo de investigación fue exitoso, ya que se pudieron aplicar técnicas o metodologías de planificación en base a tiempos y costes, para traslado de objetos de localidades en donde se emiten y receptan las cajas. Todo esto nos puede dar la pauta para el post - desarrollo de una aplicación más completa y compleja, que sirva al sector de trasporte o sector inmobiliario. Además, se observa que la planificación puede ser aplicada en muchos sectores tales como el agrícola, minero, industrial e informático, ya que se pueden modelar parámetros o factores críticos, los cuales inciden en el desarrollo de una entidad.

Por otra parte, para el desarrollo de esta tesis se tuvo que buscar analizadores que puedan compilar el código fuente, con su correspondiente planificador, modelando en su primera etapa en lenguaje natural para luego poder ser representado en PDDL, y posteriormente compilado por el analizador. PLTool fue la herramienta que se escogió, porque es un programa muy completo, que indica que planificador se puede utilizar, dando resultados en forma ordenada y con mejor presentación, en comparación con compiladores tradicionales que muestran resultados en consola. Otra ventaja es que los resultados son generados tipos de archivos XML, esto nos ayudó de gran manera, ya que se utilizó dicho fichero para realizar la simulación animada para mayor comprensión y entendimiento.

Es importante citar que los valores que representaron el problema, tales como el nivel de gasolina, distancia entre ciudades y cilindraje de los camiones, son valores a escala, es decir, representaciones mínimas del valor real, esto se realizó para un mejor manejo de la aplicación y mayor entendimiento.

Una desventaja que obtuvimos en el desarrollo del problema en lenguaje PDDL, es que los resultados no son tan comprensibles, por que se generan dichos resultados en texto plano, lo cual nos llevó a realizar una simulación animada en la que se pueda representar de forma gráfica cada estado, acción y efecto que conllevó la realización del problema de transporte de objetos.

Con respecto a los resultados obtenidos y de la aplicación por otras técnicas, resulta muy interesante analizar que CSP no se puede aplicar en nuestro problema, porque no se puede modelar factores tan decisivos en nuestro ejercicio tales como la maximización de combustible, distancia entre rutas, y cilindraje de cada automotor. Si analizamos con detenimiento, podemos observar que estos tres factores críticos están entrelazados y son dependientes uno de otro, ya que entre mayor distancia exista entre rutas, mayor consumo de gasolina, y a esto se agrega el factor cilindraje.

Por otro lado, con algoritmos genéticos se puede modelar parte de nuestro ejercicio, tratando de encontrar una ruta válida, tomando en consideración que depende mucho de la función objetivo o función *fitness* que se desarrolle para que el ejercicio pueda ejecutarse de manera satisfactoria. Con algoritmos genéticos se podría modelar los factores que se citaron en el párrafo anterior, lo que resultaría en el desarrollo de algoritmos muy complicados.

Nosotros recomendaríamos realizar el desarrollo de este o de un problema similar, en donde se consideren factores como tiempo, costo/beneficio, etc, por medio de algoritmos híbridos, los cuales son resultantes de combinar técnicas clásicas y algoritmos genéticos. Los resultados serían muy interesantes, pudiendo realizar una comparación de que técnica es más óptima y utiliza menos y mejores recursos.

Finalmente, se recomienda realizar un estudio de campo, el cual de resultados claros para poder establecer y estructurar de la mejor manera el algoritmo

híbrido, que pueda satisfacer en su totalidad al problema que se estudie, esto traerá como consecuencia un mejor modelaje de variables que pueden incidir sobre el mismo.

ANEXOS

Anexo 1: Instalación de la herramienta PLTool

Este anexo es un resumen de la instalación de la herramienta PLTool [67].

Instalación Herramienta PLTool

PLTool es una herramienta que compila código desarrollado en lenguaje PDDL, esta herramienta ofrece soporte grafico para los planificadores *PRODIGY, SAYPHI, LPG, SGLPAN y METRIC-FF*, facilitando su uso, desarrollado por el grupo de planificación y aprendizaje de la Universidad Carlos III de Madrid.

Para realizar la instalación de PLTool hay que tener algunos requerimientos tales como:

- Sistema Operativo Linux
- Compilador gcc versión 4.0 y superiores.
- Planificadores *LPG, METRIC-FF y SGPLAN*.
- Versión de la librería gtk 2.0 y superiores.

Instalación

A continuación de detalla la instalación de la herramienta:

1. Se debe crear un directorio, en el que se instalara la aplicación.
   Por ejemplo: *%mkdir pltool.*

2. Una vez creado el directorio, colocaremos adentro los ficheros *pltool.tgz* e *instal.sh.*

3. Antes de ejecutar el *.sh*, le daremos permisos de ejecución
   %chmod +x install.sh

4. Una vez asignados los permisos de ejecución, procedemos a ejecutar el archivo *install.sh*

% ./install.sh

Ejecución

Una vez instalado PLTool, se podrá arrancar la herramienta de la siguiente manera:

- El momento que se realizó la instalación se generó un archivo llamado *pltool.sh*, el cual le daremos permisos de ejecución, previamente situados en donde se encuentre éste archivo.

  chmod +x pltool.sh

- Finalmente ejecutaremos el archivo *pltool.sh*.

  ./pltool.sh

Interfaz Grafica

Al momento de arrancar la aplicación aparecerá una interfaz gráfica, en la cual podremos cargar nuestros dominios y problemas que anticipadamente habremos establecido. La herramienta consta con algunos planificadores, para nuestro problema que consta de tres etapas utilizaremos los planificadores *METRIC-FF* y *LPG*.

Planificador *METRIC-FF*

Es una extensión del planificador *FF*, desarrollado por Jörg Hoffmann, implementado en su totalidad en C. En la ilustración 54, se puede observar su correspondiente interfaz.

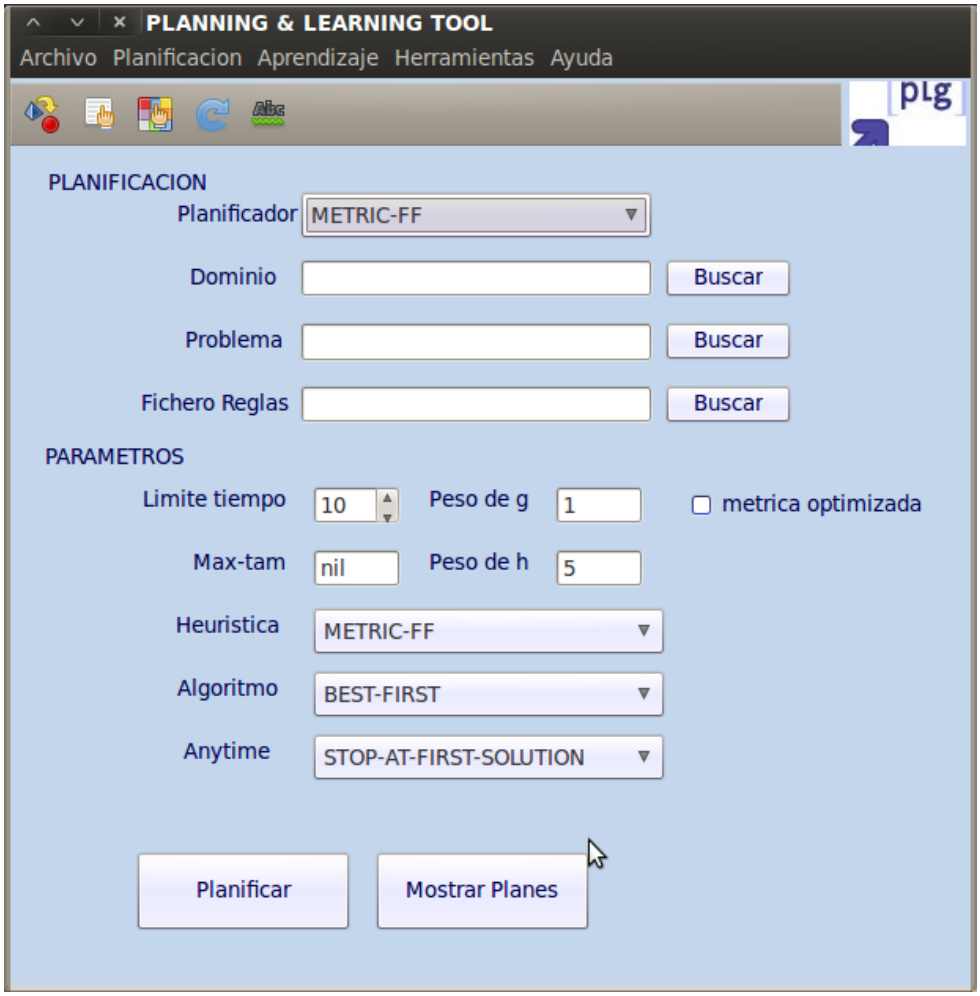


Ilustración 54. Interfaz gráfica correspondiente al planificador METRIC-FF

Planificador *LPG*

Basado en un algoritmo de mejor primero, muy similar al que usa *FF*, recomendado para dominios que tienen cantidades numéricas y duraciones. En la ilustración 55, se puede observar su correspondiente interfaz.

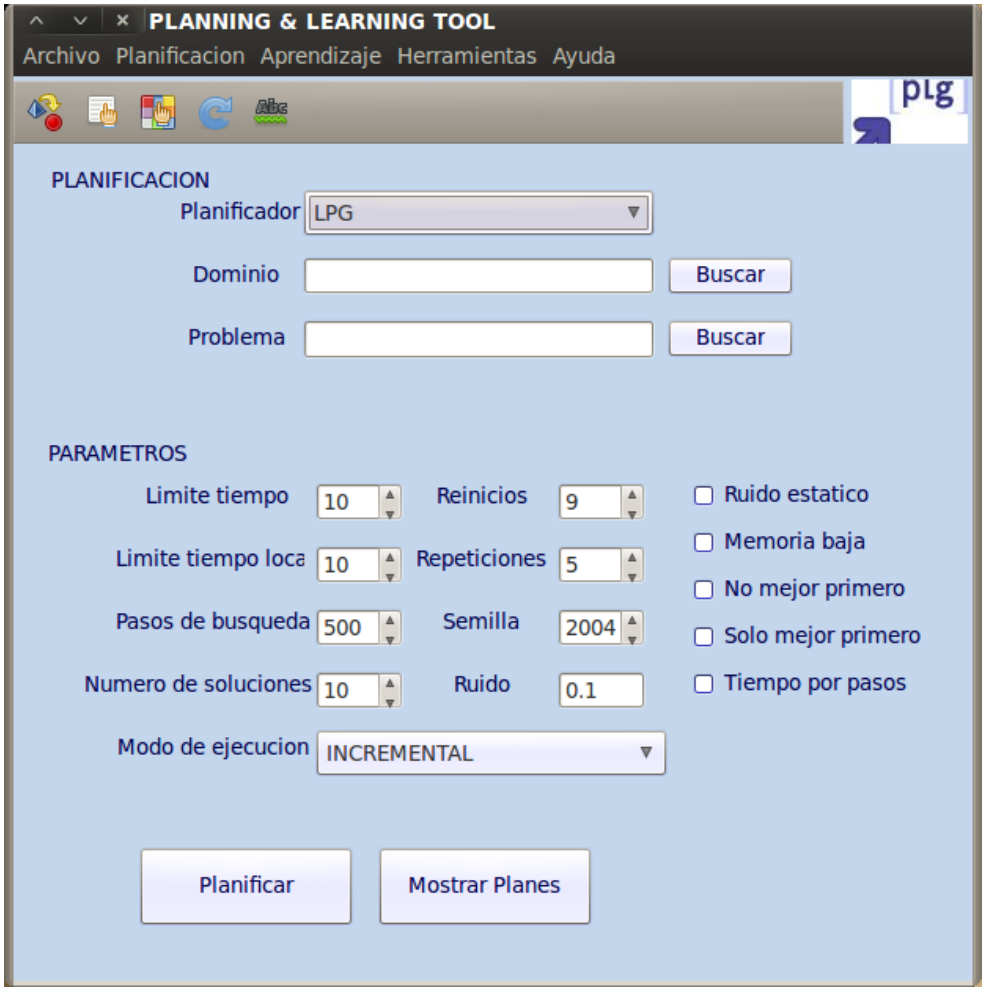


Ilustración 55. Interfaz gráfica correspondiente al planificador LPG.

Anexo 2: Simulación animada de resultados del problema

Este anexo hace referencia a parte del código fuente del programa de animación y simulación de resultados del problema de transporte de objetos.

Código fuente

En las ilustración 56, 57, 58 y 59 se puede observar la parte más relevante del código fuente correspondiente al simulador grafico de resultados.

```
public void leerXml()
   {
try {
      SAXBuilder builder = new SAXBuilder();
                  Document doc = builder.build(new
            FileInputStream("src/xml/planPrograma.xml"));
                  Element raiz = doc.getRootElement();
            System.out.println("Nombre de la raiz: "+raiz.getName());
                  List<Element> hijosRaiz = raiz.getChildren();
                  List<Element> hijos = hijosRaiz.get(0).getChildren();
            int contador=1;

            listaOperaciones = new ArrayList<ModelBase>();

            for(Element e : hijos)
                  {
            if(contador==2) break;
                     String nombre = e.getName();
            System.out.println("SUBGAFO PLAN:----------------------------------
            ----- "+nombre+"#"+contador);
            List<Element> atributos = e.getChildren();
            for(Element atributo : atributos)
                     {
                        List<org.jdom.Attribute > listAtributos =
            atributo.getAttributes();
                        String atributes=listAtributos.get(0).getValue();
                        String aux = listAtributos.get(1).getValue();
```

Ilustración 56. Primera parte del código fuente de la animación.

```
if (atributes.equals("plancost")) costo=Double.parseDouble(aux);
        if (atributes.equals("planlength")) largo=Double.parseDouble(aux);
        if (atributes.equals("totalTime")) tiempo=Double.parseDouble(aux);
                String valores=aux.substring(1,aux.length()-1);
        System.out.println("Nombre: "+atributes+"\tValor: "+valores);
        String tokens[] = valores.split(" ");
        if (atributes.equals("CARGAR-CAMION"))
        {
                ModelCargar cargar = new ModelCargar();
        cargar.setCiudad(tokens[0]);
        cargar.setVehiculo(tokens[1]);
        cargar.setObjeto(tokens[2]);
        cargar.setConductor(tokens[3]);
        listaOperaciones.add(cargar);
        }
        if (atributes.equals("TRANSPORTAR-OBJETO"))
        {
                ModelTransportar transportar = new ModelTransportar();
        transportar.setCiudadOrigen(tokens[0]);
        transportar.setCiudadDestino(tokens[1]);
        transportar.setObjeto(tokens[2]);
        transportar.setVehiculo(tokens[3]);
        listaOperaciones.add(transportar);
                }
        if (atributes.equals("DESCARGAR-OBJETO"))
                {
                ModelDescargar descargar = new ModelDescargar();
        descargar.setVehiculo(tokens[0]);
        descargar.setCiudadDestino(tokens[1]);
        descargar.setObjeto(tokens[2]);
        listaOperaciones.add(descargar);
        }
                }
        contador++;
                }
        }  catch (FileNotFoundException e) {
        e.printStackTrace();
             } catch (JDOMException e) {
        e.printStackTrace();
             } catch (IOException e) {}
```

Ilustración 57. Segunda parte del código fuente de la animación.

```
e.printStackTrace();
System.out.println("Numero de operaciones: "+listaOperaciones.size());
lblCosto.setText(costo+"");
lblLargo.setText(largo+"");
lblTiempo.setText(tiempo+"");
}

public void actionPerformed(ActionEvent e)
   {
      Object o = e.getSource();
if (o.equals(relojPrincipal1))
{
                ModelBase m = listaOperaciones.get(cont1);
        if (m instanceof ModelCargar)
                {
                   ModelCargar cargar = (ModelCargar)m;
                   ModelCiudad ciudad =
        listaCiudades.get(cargar.getCiudad());
        System.out.println("Cargar "+cargar.getVehiculo()+" en ciudad:
        "+cargar.getCiudad());
        txaPasos.append("Cargar "+cargar.getVehiculo()+" en ciudad:
        "+cargar.getCiudad()+"\n");
        if (cargar.getVehiculo().equals("CAMION1"))
        ciudad.getCamion().setIcon(imgCamion1);
        if (cargar.getVehiculo().equals("CAMION2"))
        ciudad.getCamion2().setIcon(imgCamion2);
        ciudad.getLstObjetos().remove(cargar.getObjeto());
        ciudad.getContador().setText("x"+ciudad.getLstObjetos().size())
        ;
                }
        if (m instanceof ModelTransportar)
                {
                   ModelTransportar transportar =
        (ModelTransportar)m;
                   ModelCiudad ciudadOrigen =
        listaCiudades.get(transportar.getCiudadOrigen());
                   ModelCiudad ciudadDestino =
        listaCiudades.get(transportar.getCiudadDestino());
```

Ilustración 58. Tercera parte del código fuente de la animación.

```
System.out.println("Transportar "+transportar.getVehiculo()+"
"+transportar.getCiudadOrigen()+"-
>"+transportar.getCiudadDestino());
txaPasos.append("Transportar "+transportar.getVehiculo()+"
"+transportar.getCiudadOrigen()+"-
>"+transportar.getCiudadDestino()+"\n");
if (transportar.getVehiculo().equals("CAMION1"))
{
ciudadOrigen.getCamion().setIcon(null);
ciudadDestino.getCamion().setIcon(imgCamion1);
}
if (transportar.getVehiculo().equals("CAMION2"))
{
ciudadOrigen.getCamion2().setIcon(null);
ciudadDestino.getCamion2().setIcon(imgCamion2);
            }
         }
if (m instanceof ModelDescargar)
         {
            ModelDescargar descargar = (ModelDescargar)m;
            ModelCiudad ciudad =
listaCiudades.get(descargar.getCiudadDestino());
System.out.println("Descargar "+descargar.getVehiculo()+" en ciudad:
"+descargar.getCiudadDestino());
txaPasos.append("Descargar "+descargar.getVehiculo()+" en ciudad:
"+descargar.getCiudadDestino()+"\n");
ciudad.getLstObjetos().add(descargar.getObjeto());
ciudad.getContador().setText("x"+ciudad.getLstObjetos().size());
}
cont1++;
if (cont1>=listaOperaciones.size()) relojPrincipal1.stop();
}
   }
```

Ilustración 59. Cuarta parte del código fuente de la animación.

Apéndice A

Dominios de planificación

En esta sección se detallan los dominios del problema que se busca solucionar, como ejemplo para un mayor entendimiento de los mismos.

A.1 Dominio *logistics*

Este dominio trata la problemática sobre el traslado de paquetes, ya sean por medio terrestre, cuando se encuentra en la misma localidad o ciudad, y por avión cuando se traslada los paquetes entre ciudades. Los lugares destino de estos paquetes son las oficinas de correo y aeropuertos [23].

La representación de este dominio en lenguaje PDDL se encuentra en la ilustración 60:

```
(define (domain logistics-aips98))
    (:requeriments  :strips  : equality)
    (:predicates (object ?o) (truck ¿t) (at ?0 ?1) (inside ?o ?t))
                 (airplane ?p) (airport ?s) (loc-at  ?s ?city) (city ?c)
          (location ?1))
(:action load-truck
   :parameters ( ?o ?truck ?loc)
   :precondition (and (object ?o) (truck ?truck) (lacation ?loc)
(:action load-airplane
   :paraments (?o ?p) ?loc )
                :precondition (and (object ?o) (airplane ?p)
                           (airport ?loc)  (at  ?s ?loc)  (at ?p ?loc))
                :effect (and (not (at ?o ?loc))  (inside ?o ?p)))
(:action unload-truck
   :parameters ( ?o ?truck ?loc)
                :precondition  (and  (object  ?o)  (lacation  ?loc)   (truck  ?  truck)
```

```
(inside ?o ? truck)  (at ?truck ?loc))
                    :effect (and (at ?o ?loc)) (not (inside ?o ?truck))))
                (:action unload-airplane
               :parameters ( ?o ?p ?loc)
:precondition (and (object ?o) (lacation ?loc)  (airplane ? p)    (inside ?o ? p)  (at
?p ?loc))effect (and (at ?o ?loc))
 (not (inside ?o ?p))))
                 :action fly-airplane
               :parameters ( ?p  ?s  ?d)
                    :precondition (and (airplane  ?p) (airport ?s)   (at  ? p  ?s)
                                  (not (=  ?s  ?d)))
                    :effect (and (at ?p ?d)) (not (at ?p ?s))))
                 (:action drive-truck
               :parameters ( ?truck ?s ?d ?city)
                    :precondition (and (truck  ?truck) (lacation ?s)  (location ? d)
                                  (city ?city)  (at  ?truck  ?s)  (loc-at  ?s  ?city)
                                  (loc-at  ?d  ?city)  (not (+  ?s  ?d)))
                    :effect (and (at ?truck  ?d)) (not  ?at  ?s)))))
(
```

Ilustración 60. Código fuente del problema de traslado de paquetes

A2 Dominio de mundo de bloques

Este dominio trata la problemática del traslado de bloques etiquetados (a, b, c…..), los cuales se apilen unos sobre otros, existiendo una mesa donde colocarlos y un brazo que ejecutará las acciones para solucionar este problema [25].

La representación de este dominio en lenguaje PDDL es la siguiente:

```
(define (domain mundobloques))
            :requirements  :strips   :equality)
:predicates   (encima   ?b11   ?b12)
                          (enmesa  ?b1)
                          (libre   ?b1)
                          (sujeto   ?b1)
                          (brazolibre)
)
            (:action quitar
                 :parameters  (?b11  ?b12)
                 :precondition (and   (encima   ?b11   ?b12))
                                (libre  ?b11 )
                                (brazolibre)
                                 )
                 :effect   (and   (sujeto  ?b11)
                                (not (encima  ?b11  ?b12  ) )
                                (not (brazolibre) )
                                (not (libre ?b11  ))
                              )
              )
              (:action levanter
                 :parameters  (?b11)
:precondition (and   (en mesa   ?b11 ))
                                (libre  ?b11 )
                                (brazolibre)
)
                 :effect   (and   (sujeto  ?b11)
                                (not (enmesa  ?b11 ) )
                                (not (brazolibre) )
                                (not  (libre ?b11  ))
                              )
             )
              (:action poner
                 :parameters  (?b11   ?b12)
                 :precondition (and   (sujeto  ?b11 ))
 (libre  ?b12 )
                                 )
                 :effect   (and   (encima  ?b11  ?b12 )
(libre  ?b11 )
(brazolibre)
(not  (sujeto ?b11  ))
       (not  (libre ?b12  ))
                              )
             )
              (:action dejar
:parameters  (?b11)
                 :precondition (and   (sujeto   ?b11 )
)
```

```
                :effect   (and   (enmesa  ?b11)
(libre  ?b11)
(brazolibre)
 (not (sujeto ?b11  ))
                              )
              )
)
         (define   (problem ejemplo)
                   (:domain   mundobloques)
                   (:objects   a  b  c  d )
                       (:init (enmesa     d)
(enmesa     c)
                                  (libre         a)
                                  (libre         c)
(encima   a   b)
(encima   b   d)
(brazolibre))
                     (:goal       (and
                                        (enmesa     a)
                                        (enmesa     b)
  (encima   c   b)
                                        (encima   d   c)
                                        (libre         a)
                                        (libre         d)
                                   ) )
```

Ilustración 61. Código fuente correspondiente al traslado de bloques

Apéndice B

B1 Problema del viajante del comercio

Problema que representa a un viajero que se encuentra en una ciudad o localidad, llamado punto inicial y que quiere desplazarse por *n* ciudades, sin pasar por la misma ciudad dos veces. Se debe minimizar la distancia total recorrida, teniendo como cometido encontrar la ruta más óptima, la siguiente ilustración 62 representa dicho ejemplo.

.

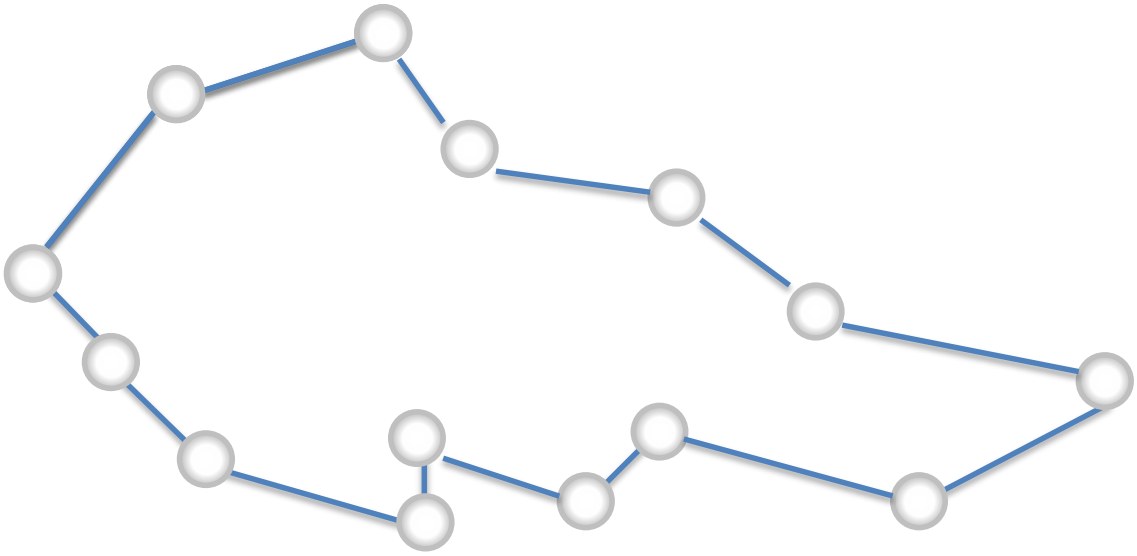


Ilustración 62. Problema del viajante del comercio

BIBLIOGRAFIA

[1] RUIZ REINA José L, Planificación, Dpto. de Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla, Año 2011, fecha de recuperación: 10-Feb-2012, http://www.cs.us.es/cursos/ia1/temas/tema-08.pdf

[2] ORTEGA Gonzalo Martin y GARCIA Carlos Martin. Estudio de algoritmos para la planificación en I.A, Universidad Politécnica de Salamanca, Campus de Madrid, fecha de recuperación: 11-Feb-2012, http://www.negomobile.es/sites/default/files/data/proyectos/Shop/EstudioTecnico.pdf

[3] ALER MUR Ricardo, Programación Genética de Heurísticas para Planificación, Universidad Politécnica de Madrid, Año 1999, fecha de recuperación: 11-Feb-2012, http://oa.upm.es/1101/1/10199907.pdf

[4] IA.pdf Notas de inteligencia Artificial reconocimiento de formas e imagen digital, Universidad Politécnica de Valencia, Ano 2009, fecha de recuperación: 12-Feb-2012, www.dsic.upuv.es

[5] JIMENEZ LINARER Luis y SANCHEZ CRESPO Luis Enrique, Sistemas Basados en el conocimiento (Planificación), Inteligencia Artificial e Ingeniería del Conocimiento, Universidad de Castilla – La Mancha, fecha de recuperación: 13-Feb-2012, http://www.google.com.ec/url?sa=t&rct=j&q=JIMENEZ+LINARER+Luis%2C+SANCHEZ+CRESPO+Luis+Enrique%2BSistemas+Basados+en+el+conocimiento+(Planificaci%C3%B3n)%2BUniversidad+de+Castilla+&source=web&cd=2&cad=rja&ved=0CCUQFjAB&url=http%3A%2F%2Fwww.sanchezcrespo.org%2FDocencia%2FIA%2FIA%2520-%2520Tema%25204E%2520-%2520Sistemas%2520Expertos%2520v1.1.ppt&ei=hK1HUN_EOoLe8wTAvIH4Cg&usg=AFQjCNGubPGAhmcUjg4PUeFyg9VWFObrow

[6] TURBIDEZ Tomas de la Rosa, Razonamiento Basado en Casos Aplicado a la Planificación Heurística, Universidad Carlos III de Madrid, Año 2009, fecha de recuperación: 20-Feb-2012, http://e-archivo.uc3m.es/bitstream/10016/9713/1/Tesis_TdelaRosa_CBR.pdf

[7] Inteligencia Artificial, Técnicas de Búsquedas Heurísticas, fecha de recuperación: 23-Feb-2012, http://subversion.assembla.com/svn/ii_0910/09-10/IA1/Teor%C3%ADa/Tema_3_-_Busqueda_heuristica.pdf

[8] "Búsqueda Heurística", Sistemas expertos Inteligencia Artificial, fecha de recuperación: 26-Feb-2012,

http://old.udb.edu.sv/Academia/Laboratorios/informatica/Sistemas%20Expertos/guia4SEeIA.pdf

[9] MESEGUER Pedro, Búsqueda Heurística I, IIIA-CSIC, Bellaterra España, fecha de recuperación: 27 Feb-2012, http://www.iiia.csic.es/~pedro/busqueda1-introduccion.pdf

[10] MALAGON Constantino, Búsqueda Heurística, Área de computación e Inteligencia Artificial, fecha de recuperación: 2-Mar-2012, http://www.nebrija.es/~cmalagon/ia/transparencias/busqueda_heuristica.pdf

[11] GARCIA PEREZ Oscar J, Planificación en Dominios Temporales usando técnicas HTN, Departamento de Ciencias de la Computación e Inteligencia Artificial, fecha de recuperación: 8-Mar-2012, http://hera.ugr.es/tesisugr/16795143.pdf

[12] LSI- FIB- UPS, Introducción a la Planificación, Curso 2011/2012, fecha de recuperación: 10-Mar-2012 http://www.lsi.upc.edu/~jvazquez/teaching/iag/transpas/4-PL1-IntroPlanificaci%C3%B3n.pdf

[13] FERNANDEZ REBOLLO Fernando, Desarrollo y Selección de características basadas en distancias entre grafos para problemas de producción en Planificación Automática, Departamento de Informática, Universidad Carlo III de Madrid, Septiembre 2011, fecha de recuperación: 11-Mar-2012 http://e-archivo.uc3m.es/bitstream/10016/13018/1/PFC_IR_Cenamor Guijarro.pdf

[14] Ingeniería Artificial e Ingeniería del Conocimiento, Planificación, fecha de recuperación: 15-Mar-2012
Web: http://decsai.ugr.es/iaic/Curso2008-09/programa-iaic-2008-09.htm

[15]Inteligencia Computacional, Inteligencia Artificial Explicada, fecha de recuperación: 16-Mar-2012, http://www.smartcomputing.com.ar/representacion-mediante-strips.aspx

[16] BASTERRICA BROCKMAN Daniel, Proyecto de investigación aplicada: CSP, Inteligencia Artificial, Año 2004, fecha de recuperación: 16- Jun-2012, http://www.alumnos.inf.utfsm.cl/~ntroncos/files/csp.pdf

[17] LOPEZ Bruno, Algoritmo Hill Climbing, fecha de recuperación: 20-Mar-2012, http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Inteligencia%20Artificial/Apuntes/IA/Hill-Climbing.pdf

[18] GUERVOS MERELO Juan Julián, Técnicas Heurísticas de Resolución de Problemas: Computación Evolutiva y redes neuronales, fecha de recuperación: 21-Mar-2012 http://geneura.ugr.es/~jmerelo/tutoriales/heuristics101/

[19] Introducción a la Inteligencia Artificial, Búsqueda con Información, Presentación de Microsoft PowerPoint97-2003, fecha de recuperación: 21-Mar-2012 http://www.google.com.ec/url?sa=t&rct=j&q=b%C3%BAsqueda%20con%20informaci%C3%B3n%2Bmarvin%20minsky&source=web&cd=10&sqi=2&ved=0CFwQFjAJ&url=http%3A%2F%2Fdsi.fceia.unr.edu.ar%2Fdownloads%2FIIA%2Fpresentaciones%2Fbusqueda2-2007.ppt&ei=x7-9T6yoF8fxggfGutHvDQ&usg=AFQjCNEU7axncJCfcFz9EbFF_1IB4hko7g

[20] FUENTETAJA PIZAN Raquel, Tesis doctoral, Búsqueda Heurística en Planificación Basada en Costes, Departamento de Informática, Universidad Carlos III de Madrid, Leganés, Junio 2010, fecha de recuperación: 23-Mar-2012 http://scalab.uc3m.es/~clinares/download/rfuentetaja/rfuentetaja.doc.pdf.gz

[21] RUIZ REINA José, ALONSO José Antonio y MATEOS MARTIN Francisco J, Búsqueda informada mediante Técnicas Heurísticas, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla, 2011, fecha de recuperación: 24-Mar-2012, http://www.cs.us.es/cursos/ia1/temas/tema-04.pdf

[22] RIVERA LOPEZ Rafael, Inteligencia Artificial I, Departamento de Sistemas y Computación, Veracruz Ago-Dic 2008, fecha de recuperación: 25-Mar-2012, http://iscitver2011.files.wordpress.com/2011/02/4-2sat-restricciones.pdf

[23] FERNANDEZ ARREGUI Susana, Aprendizaje de Conocimiento de Control para Planificación de Tareas, Departamento de Informática, Universidad Carlos III de Madrid, Escuela Politécnica Superior, Leganés 2006, fecha de recuperación: 28-Mar-2012, http://e-archivo.uc3m.es/bitstream/10016/656/1/Tesis_Susana_Fernandez_Arregui.pdf

[24] PESQUEREA MARTIN Alberto, Aprendizaje en Inteligencia Artificial, fecha de recuperación: 30-Mar-2012, http://sindominio.net/~apm/articulos/IAIC/aprendizaje/aprendizaje.pdf

[25] RIBADAS PENA Francisco José, Aprendizaje Deductivo, Mayo-2008, fecha de recuperación: 2-Abr-2012, http://ccia.ei.uvigo.es/docencia/MRA/T3.pdf

[26] HAMMOND Kristian J, Un modelo de Planificación Basado en Casos, Departamento de Ciencias de la Computación, Universidad Yale, fecha de recuperación: 2-Abr-2012, https://www.aaai.org/Papers/AAAI/1986/AAAI86-044.pdf

[27] GARRILLO Antonio y MORALES Lluvia, E-learning y Planificación Inteligente: Mejorando la Personalización de Recursos, Universidad Politécnica de Valencia, Universidad Tecnológica Mixteca, fecha de recuperación: 4-Abr-2012, http://users.dsic.upv.es/~agarridot/index_archivos/papers/garrido12.pdf

[28] GONZALEA GARCIA Fernando, Introspección en sistemas de Planificación basada en casos aplicados a juegos de estrategia, Facultad de Informática, Universidad Complusente de Madrid, Septiembre 2011, fecha de recuperación: 5-Abr-2012, http://eprints.ucm.es/13506/1/Introspecci%C3%B3n_en_sistemas_de_planificaci%C3%B3n_basada_en_casos_aplicados_a_juegos_de_estrategia.pdf

[29] OLLAVE CARRANZA Arturo Demetrio y JIMENEZ BUILES Jovani Alberto, Educación Virtual-Evaluación, Universidad Nacional de Colombia sede Medellín, fecha de recuperación: 7-Abr-2012, http://www.colombiaaprende.edu.co/html/mediateca/1607/articles-108458_archivo.pdf

[30] ARIAS SANCHEZ Francisco Javier, Modelo Multi-Agente para la Planificación Instruccional y Selección de Contenidos en Cursos Virtuales Adaptativos, Facultad de Minas, Escuela de Sistemas, Universidad Nacional de Colombia sede Medellín, 2009 fecha de recuperación: 7- Abr-2012, https://docs.google.com/viewer?a=v&q=cache:q0mO-Z8W2n8J:www.bdigital.unal.edu.co/1998/1/18010764.20101.pdf+ARIAS+SANCHEZ+Francisco+Javier,+Modelo+Multi-Agente+para+la+Planificaci%C3%B3n+Instruccional+y+Selecci%C3%B3n+de+Contenidos+en+Cursos+Virtuales+Adaptativos&hl=es-419&gl=ec&pid=bl&srcid=ADGEESiRibhdMWzOrxl50yvnTzMFO_6SeeJBNGKhX8nJVhUSzvNnd8_zWH_9SDuNX_4frOV6B1h4zjqQyf4sUnWpfcFNF9lcatEo15EtMXD9T9-bE4PxQuKw7DlBrqejTZU2_DuABpqI&sig=AHIEtbSLG3PN1cakAQt4SIc3w6_qfPr1XA

[31] ARIAS Francisco, JIMENEZ Jovani y OVALLE Demetrio, Modelo de Planificación Instruccional en Sistemas Tutoriales Inteligentes, Escuela de Ingeniería de Sistemas, Universidad Nacional de Colombia sede Medellín, Mayo-2009, fecha de recuperación: 8- Abr-2012, http://www2.unalmed.edu.co/~pruebasminas/index.php?option=com_docman&task=doc_view&gid=563&tmpl=component&format=raw&Itemid=285

[32] RAMOS Silvia, Heurísticas y problemas Combinatorios, fecha de recuperación: 10-Abr-2012http://materias.fi.uba.ar/7114/Docs/ApunteHeuristicas.pdf

[33] Grupo de Planificación y Aprendizaje, Planificación, Departamento de Informática, Universidad Carlos III de Madrid, Diciembre 2008, fecha de recuperación: 11- Abr-2012, http://ocw.uc3m.es/ingenieria-informatica/inteligencia-artificial-2/material-de-clase-1/planificacion.pdf

[34] CENAMOR GUIJARRO Isabel Rosario, Desarrollo y Selección de Características Basadas en Distancias Entre Grafos Para Problemas de Predicción en Planificación Automática, Departamento de Informática, Universidad Carlos III de Madrid, Septiembre 2011, fecha de recuperación: 13- Abr-2012, http://e-archivo.uc3m.es/bitstream/10016/13018/1/PFC_IR_Cenamor_Guijarro.pdf

[35] GARRIDO Antonio, ONAINDIA Eva y BARBER Federico, Integración de Planificación Temporal y Abstracción de Recursos, Dpto. de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, fecha de recuperación: 15- Abr-2012,
http://www.google.com.ec/url?sa=t&rct=j&q=GARRIDO+Antonio%2C+ONAINDIA+Eva+y+BARBER+Federico%2C+Integraci%C3%B3n+de+Planificaci%C3%B3n+Temporal+y+Abstracci%C3%B3n+de+Recursos&source=web&cd=2&cad=rja&ved=0CCUQFjAB&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.78.9328%26rep%3Drep1%26type%3Dps&ei=MrlHUK67EZL68QScm4A4&usg=AFQjCNHMYsH-FS6I2UjXix5k0Db_S0lbUw

[36] LOPEZ DIAZ José Carlos, Un algoritmo genético con codificación real para la evolución de transformaciones lineales, Departamento de Informática, Universidad Carlos III de Madrid, Leganés Julio 2010, fecha de recuperación: 16- Abr-2012, http://e-archivo.uc3m.es/bitstream/10016/10214/1/PFC_JOSE_CARLOS_LOPEZ_DIAZ.pdf

[37] GRUPO PLG, Planificación Automática, Universidad Carlos III de Madrid, 2008-2009, fecha de recuperación: 23- Abr-2012, http://ocw.uc3m.es/ingenieria-informatica/planificacion-automatica/transparencias/conocimiento-control.pdf

[38] REYES BALLESTEROS Alberto, Representación y Aprendizaje de Procesos de Decisión de Markov Cualitativos, Instituto Tecnológico y de Estudios Superiores de Monterrey, Cuernavaca Noviembre 2006, fecha de recuperación: 25- Abr-2012, http://campus.cva.itesm.mx/areyes/reportes/tesisARB.pdf

[39] FERNANDEZ Fernando, BORRAJO Daniel y LINARES Carlos, Planificación con Incertidumbre, Departamento de Informática, Escuela Politécnica Superior, Universidad Carlos III de Madrid, Febrero-2007, fecha de recuperación: 26- Abr-2012, http://www.plg.inf.uc3m.es/~pad/06-07/transparencias/planificacion
_incertidumbre2.pdf

[40] TORRES ANDRADE Cesar y CAVIEDES Diego, Toma de decisiones Complejas fecha de recuperación: 27- Abr-2012,
http://diegocaviedes.host56.com/docs/Capitulo_17.pdf

[41] REBOLLO FERNANDEZ Fernando y MILLAN BORRAJO Daniel, Procesos de Decisión de Markov Aprendizaje Automático, Departamento de Informática, Escuela Politécnica Superior, Universidad Carlos III de Madrid, Febrero 2009, fecha de recuperación: 28- Abr-2012, http://ocw.uc3m.es/ingenieria-informatica/aprendizaje-automatico/material-de-clase-1/aa-ocw-mdp.pdf

[42] MORALES REYNAGA Lluvia C, Generación Automática de Diseños de Aprendizaje: Diferentes enfoques de planificación, Universidad de Granada, Mayo – 2011, fecha de recuperación: 28- Abr-2012 http://buhoz.net/public/lluvia/documents/TesisLluviaFinal.pdf

[43] PALMA DURAN Juan R, Extendiendo Darmok, Un sistema de Planificación Basada en Casos, mediante arboles de Comportamiento", Facultad de Informática, Universidad Complutense de Madrid, Curso 2009/2010, fecha de recuperación: 2- May-2012, http://eprints.ucm.es/11433/1/Memoria.pdf

[44] GALINDO Cipriano, FERNANDEZ Juan y GONZALEZ Javier, Planificación de Tareas en entornos modelados con jerarquías de abstracción, Departamento de Ingeniería en Sistemas y Universidad de Málaga, 2002, fecha de recuperación: 4- May-2012, http://search.conduit.com/Results.aspx?q=GALINDO+Cipriano%2C+FERNANDEZ+Juan+y+GONZALEZ+Javier%2C+Planificaci%C3%B3n+de+Tareas+en+entornos+modelados+con+jerarqu%C3%ADas+de+abstracci%C3%B3n%2B2002&Suggest=&stype=Homepage&SelfSearch=1&SearchType=SearchWeb&SearchSource=10&ctid=CT3148764&octid=CT3148764

[45] Introducción a la Inteligencia Artificial, fecha de recuperación: 6- May-2012, http://dmi.uib.es/~abasolo/intart/1-introduccion.html

[46] RUSSELL Stuart y NORVING Peter, Planificación, Inteligencia Artificial un Enfoque Moderno, fecha de recuperación: 7- May-2012, http://ca.inf.udec.cl/files/inteligencia_artificial/LibroGuia/CAP11%20-%20Planificaci%F3n.pdf

[47] Búsqueda Heurística, fecha de recuperación: 10- May-2012, http://www.infor.uva.es/~arancha/IA/busqueda/busq2.pdf

[48] CARRILLO SEGUNDO Pablo, Ontología para Planificación Lineal, Universidad Politécnica de Madrid, Ano 2002, fecha de recuperación: 12- May-2012, http://oa.upm.es/224/1/05200217.pdf

[49] BROMBEG Facundo, Búsqueda Informada, Abril-2008, fecha de recuperación: 15-May-2012, http://ai.frm.utn.edu.ar/fbromberg/iaa-uncu-2008/lectureNotes/2008-10-01-busqueda-informada.pdf

[50] PACHECO Alberto, Búsquedas Heurísticas, Marzo-1999, fecha de recuperación: 17- May-2012, http://www.depi.itch.edu.mx/apacheco/ai/busqheur.htm

[51] GALINDO GONZALEZ Carlos, PEREZ VAZQUEZ Ramiro, Búsqueda heurística, Septiembre 2009, fecha de recuperación: 18- May-2012,

http://www.gestiopolis.com/administracion-estrategia/metodos-heuristicos-en-software.htm

[52] Introducción a los algoritmos genéticos: cómo implementar un algoritmo genético en java, fecha de recuperación: 20- May-2012, http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jgap

[53] ESCOLANO RUIZ Francisco, CAZORLA QUEVEDO Miguel, ALFONSO GALIPIENSO Isabel, PARDO Otto y LOZANO ORTEGA Miguel, Modelos, Tecnicas y Áreas de Aplicación, fecha de recuperación: 19- May-2012, http://books.google.com.ec/books?id=_spC6S7UfZgC&printsec=frontcover#v=onepage&q&f=false

[54] REYES BALLESTEROS Alberto, Estado del Arte en planificación con Incertidumbre, Cuernavaca Abril 2003, fecha de recuperación: 20- May-2012 http://campus.cva.itesm.mx/areyes/reportes/EstadoArteFinal.pdf

[55] BROMBEG Facundo, Planificación, Facultad Regional Mendoza, Universidad Tecnológica Nacional, Abril 2012, fecha de recuperación: 23- May-2012 http://dharma.frm.utn.edu.ar/cursos/ia/teoricas/Teorica_U3_A_Planning/output/U3-A-capitulo11-Planning-handouts.pdf

[56] GUZMAN LUNA Jaime A, AREIZA ZABALA Daniel y ORJUELA DUARTE Ailin, Lenguaje bajado en XML para la definición de dominios de planificación orientado a la web, Escuela de Sistemas, Universidad Nacional de Colombia, fecha de recuperación: 23- May-2012, http://www.unipamplona.edu.co/unipamplona/portalIG/home_10/recursos/general/documentos/pdf/15102009/02_ar_ailin_orj.pdf

[57] SANCHEZ Antonio, Una aproximación odontológica al modelado de conocimiento en los dominios de planificación, Departamento de Ingeniería del Software e Ingeniería Artificial, Facultad de Informática, Universidad Completense de Madrid, Abril 2010, fecha de recuperación: 25- May-2012, http://eprints.ucm.es/11729/1/T32517.pdf

[58] MARTINEZ CALVINO Aida, Cooperación en los problemas del viajante (TSP) y de rutas de vehículos (VRP): una panorámica, Junio 2011, fecha de recuperación: 28-May-2012, http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_762.pdf

[59] MONTANA José L y GONZALEZ Inés, Búsqueda Local, Inteligencia Artificial e ingeniería del Conocimiento, fecha de recuperación: 29- May-2012, http://www.matesco.unican.es/informacion_academica/2006_2007/informatica/InteligenciaArtificial/b%C3%BAsqueda-local.pdf

[60] RUBIO Julio, MURO Pedro y BANARES José, Búsqueda, Departamento de Informática e Ingeniería en sistemas, Universidad de Zaragoza, Noviembre 1998, fecha de recuperación: 30- May-2012, http://www.lsi.upc.edu/~bejar/iia/iabusq.pdf

[61] Grupo PLG, Inteligencia Artificial, Universidad Carlos III de Madrid, Ano 2008, fecha de recuperación: 1- Jun-2012, http://ocw.uc3m.es/ingenieria-informatica/inteligencia-artificial/mis-items/material-de-clase-2/busqueda-ocw.pdf

[62] Fecha de recuperación: 1- Jun-2012, http://blog.vidasconcurrentes.com/programacion/busqueda-en-profundidad-y-busqueda-en-anchura/

[63] VALERO CUBAS Soledad, ARGENTE VILLAPLANA Estefanía y ONAINDIA DE LA RIVAHERRERA Eva, Proceso de paralelización y Optimación de Planes Secuenciales, Paralelos y Temporales, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Enero 2004, fecha de recuperación: 3- Jun-2012, http://www.dsic.upv.es/docs/bib-dig/informes/etd-01282004-140441/InformeInvestigacionPlanificacion.pdf

[64] BROMBEG Facundo, Planificación, Facultad Regional Mendoza, Universidad Tecnológica Nacional, Abril-2012, fecha de recuperación: 5- Jun-2012, http://dharma.frm.utn.edu.ar/cursos/ia/teoricas/Teorica_U3_A_Planning/output/U3-A-capitulo11-Planning-handouts.pdf

[65] FERNANDEZ REBOLLO Fernando, Aprendizaje por Refuerzo, fecha de recuperación: 6- Jun-2012, http://www.plg.inf.uc3m.es/~aa/transpas05-06/mdp.

[66] SUCAR Enrique, Métodos de Inteligencia Artificial, Tecnologías de información, fecha de recuperación: 7- Jun-2012, UPAEP, http://search.conduit.com/?SearchSource=10&ctid=CT3148764

[67] Manual de Usuario PLTool, Universidad Carlos III de Madrid, Leganés, fecha de recuperación: 10- Jun-2012, Noviembre-2007, http://www.plg.inf.uc3m.es/software/pltool/manual-pdf/manual-es.pdf

[68] Hernan, Algoritmo A* para encontrar el camino más corto en IA, fecha de recuperación: 12- Jun-2012, http://www.cristalab.com/tutoriales/algoritmo-a-para-encontrar-el-camino-mas-corto-en-ia-c91591l/

[69] Algoritmos Genéticos, fecha de recuperación: 13- Jun-2012, http://di002.edv.uniovi.es/~alguero/eaac/eaac_archivos/09-10/Trabajos%20para%20evaluaci%C3%B3n/Quintairos/Art%C3%ADculos%20proporcionados/temageneticos.pdf

[70] Algoritmos Genéticos, Centro de Inteligencia Artificial, Universidad de Oviedo, fecha de recuperación: 14- Jun-2012,http://www.aic.uniovi.es/ssii/SSII-T4-AlgoritmosGeneticos.pdf

[71] Fecha de recuperación: 15- Jun-2012, Web: http://dharma.frm.utn.edu.ar/cursos/ia/2012/material/APUNTES-FILMINAS/U2/Soluciones%20CSP.pdf