*Una heurística diferente*

Siamion Danko   
*dpto. Ciencias de la Computación e Inteligencia Artificial*  
*Universidad de Sevilla*Sevilla, España  
siadan@alum.us.es

Servando Figueroa Gómez  
*dpto. Ciencias de la Computación e Inteligencia Artificial*  
*Universidad de Sevilla*Sevilla, España  
serfiggom@alum.us.es

El trabajo consiste en la realización de una nueva heurística diferente basada en la obtención de una lista con los operadores necesarios, teniendo en cuenta solo los efectos positivos, para la realización del problema desde el objetivo hasta el estado inicial.

El objetivo principal del trabajo es realizar esta nueva función heurística y, en la medida de lo posible, probarla y ver su funcionamiento. Esta función deberá dar a elegir si se quiere usar o no esta nueva heurística.

Heurística: es vista como el arte de inventar por parte de los seres humanos, con la intención de procurar estrategias, métodos, criterios, que permitan resolver problemas a través de la creatividad, pensamiento divergente o lateral.

PDDL: (Lenguaje de Definición de Dominios de Planificación, en español) se ha convertido en el lenguaje estándar para describir problemas de planificación. El objetivo de este proyecto es crear una aplicación que facilite al usuario la definición de dominios y problemas de planificación.

Inteligencia artificial: Programa de computación diseñado para realizar determinadas operaciones que se consideran propias de la inteligencia humana, como el autoaprendizaje.

Operador: está formado por las precondiciones y efectos de una acción.

Objetivo: estado final del problema.

# Introducción

La **Inteligencia Artificial** es la combinación de algoritmos **planteados con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano.** Una tecnología que todavía nos resulta lejana y misteriosa, pero que desde hace unos años está **presente en nuestro día a día a todas horas [1].**

**Entre los estudios de estos algoritmos se encuentra PDDL que consiste en una representación específica para problemas de planificación [2]. La planificación consiste en encontrar una secuencia de acciones que alcanzan un determinado objetivo si se ejecutan desde un determinado estado inicial. Para este tipo de problemas se pueden llevar a cabo las heurísticas. Esto es una estrategia que acelere o haga más eficaz un problema haciendo que tome menos posibilidades descartando las peores.**

La idea del trabajo es realizar una heurística diferente, basada en una lista para dar una aproximación de cuál puede ser una solución más acertada, ver cual posibilidad llega más rápido a la finalización del ejercicio o, simplemente, intentar agilizar la solución del problema deseado. Esta heurística consistirá en una lista en la que se irá guardando los operadores necesarios (teniendo en cuenta solo los efectos positivos de los mismos) para satisfacer el objetivo. Es decir, esta operación se desarrolla del final del problema al principio, como si de una resolución del ejercicio del Algoritmo POP se tratase, aunque solo teniendo en cuenta los efectos positivos.

El uso de una **heurística** puede ayudar a agilizar un problema, procederemos a ver la importancia de estas estrategias [3]. Como se recoge en la información referenciada, al haber diversas estrategias se producen diferentes serializaciones. Se puede producir que, en el peor de los casos, la planificación busque la serialización completa. Por esto, podemos decir que, al haber menor serialización, más eficaz será nuestra estrategia. Y, además, podemos concluir con que una buena heurística dará lugar a un menor nivel de alternativas.

Se procede a introducir el funcionamiento de la heurística. Esta estrategia que se seguirá será una función recursiva en la que el objetivo irá cambiando para ir satisfaciendo los efectos positivos hasta llegar al inicio. Al ser recursivo, debe tener al menos un resultado que haga parar al problema. Según la estrategia, podemos entender dos condiciones de parada.

La primera condición de parada es si el estado inicial satisface directamente al estado objetivo, por lo tanto, se puede decir que el comienzo es igual que el final y, el problema no tendría que comenzar dado que ya ha finalizado sin empezar.

La segunda condición se da si el estado inicial no satisface directamente al objetivo y, además, no hay ningún efecto positivo de algún operador que lo satisfaga. Entonces, podemos decir que el problema es infinito y no tiene solución.

La tercera y última condición se da si hay algún operador que satisfaga al objetivo, se guarda en la lista solución y se llama de manera recursiva a la función, pero ahora el nuevo objetivo es la precondición positiva del operador que satisfacía al objetivo.

Se irán realizando tantas llamadas recursivas como sean necesarias hasta llegar al estado de parada que sería que el estado inicial sea el objetivo. Esto devolvería una lista, de la cual se necesita la longitud de esta.

A continuación, la estructura del documento. Se comienza con unos preliminares donde se especifican los métodos empleados y el trabajo relacionado. Seguido, la metodología y los resultados de las pruebas realizadas. Por último, una conclusión donde se hablará de los resultados obtenidos y los posibles cambios de mejora que se podrían llevar a cabo.

# Preliminares

A continuación, se comentarán los métodos empleados y trabajos relacionados.

## Métodos empleados

Para comenzar se tratará la búsqueda en general. Cuando se habla de búsqueda se refiere a encontrar una solución recorriendo el árbol que crea el algoritmo de PDDL. Este árbol comprende todas las posibilidades a las que se puede viajar desde un estado, es decir, empezando en un nodo estado inicial, éste tendrá varios hijos que son los posibles cambios de estado. Un hijo nunca será hijo de un padre del cual no se puede llegar desde el estado padre a él.

Sabiendo esto, podemos introducir los diferentes tipos de búsquedas, empezando por la búsqueda en profundidad. Con este tipo de búsqueda lo que se consigue es ir llegando hasta los nodos hoja, es decir, los hijos del árbol que no tienen más hijos, lo que podemos denominar el final del árbol. Como su nombre indica, este tipo de búsqueda recorre primero todos los nodos que lleven al final, da una solución más rápida, pero no tiene por qué ser la más corta.

Por otro lado, está la búsqueda por anchura. Al contrario que el de profundidad, este tipo de búsqueda pretende recorrer primero todos los hijos del padre que se está recorriendo para buscar la solución. En este caso, es más costoso encontrar la solución, pero suele ser más corta que la búsqueda anterior.

Estos dos métodos anteriores no necesitan heurística para realizarse. Estos métodos se han usado para comparar el tiempo de realización de pruebas con y sin heurística. Ahora, pasaremos a introducir los métodos que usan heurística, como puede ser la búsqueda del primero el mejor o el algoritmo A\* (A estrella).

El algoritmo de primero el mejor lo que hace es ordenar la lista de valores de los nodos del árbol para encontrar la solución, ahorrando evaluaciones a la función [4]. Esta función está definida como ayuda de 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.

Antes de pasar con el algoritmo desarrollado, se introducirá el algoritmo A estrella [5]. Este algoritmo puede ser empleado para calcular el camino mínimo en una red. Como ya se dijo, es un algoritmo heurístico, ya que su principal característica es que hará uso de una función heurística, mediante la cual irá etiquetando a los nodos de la red y que servirá para determinar la probabilidad de dichos nodos de permanecer al camino óptimo.

Por último, se introducirá la nueva búsqueda que se ha realizado, la cual se comentó por encima en la introducción. Esta búsqueda ha sido denominada como “BúsquedaConHeurística” que hace uso de la función pregoGeneral, la función que se encarga de etiquetar a los diferentes nodos. Esta función, como ya se dijo, consiste en la creación de una lista en la que se añaden los operadores que sean necesarios para alcanzar el objetivo. La etiqueta del nodo correspondiente será el número de operadores que hagan falta utilizar para llegar al estado final.

## Trabajo Relacionado

Es interesante tratar el problema de la rueda pinchada o de la rueda de repuesto [2] porque es uno de los problemas que se han empleado para probar esta nueva heurística en los casos de pruebas.

Este problema consiste en, como su nombre indica, un coche al cual se le pincha una rueda y se quiere cambiar esa rueda por una de repuesto. Como se están tratando problemas de planificación **PDDL**,la solución al problema será aquella secuencia de operadores o acciones que consigan llevar el estado inicial al estado objetivo.Esto es un problema sencillo y lógico de visualizar. Es lógico que, para poner una rueda nueva, debemos haber quitado la pinchada y que para ponerla debemos haberla sacado del maletero, que es donde estaría guardada. Como se puede observar con estas indicaciones, este problema constaría de cuatro operadores:

* Sacar: para poder sacar una rueda primero debe estar en el maletero. El efecto que produce es mover la rueda del maletero al suelo.
* Quitar: para poder quitar la rueda pinchada debe estar en el eje del coche. Esto provoca que la rueda pinchada ya no esté en el eje y pase a estar en el suelo.
* Poner: primero se debe tener la rueda de repuesto en el suelo y se tiene que haber quitado la rueda pinchada del eje para así poner la rueda nueva.
* Guardar: para guardar una rueda primero debe estar en el suelo y no debe estar la de repuesto en el maletero. El efecto que provoca es que la rueda pinchada quede guardada en el maletero, dejando de estar en el suelo, obviamente.

Con este problema se han implementado algunas pruebas para ver el correcto funcionamiento de la heurística y comprobar los tiempos de rendimiento.

Como conclusión a este problema, es un problema sencillo para poder comprobar a mano la heurística y poder ver si está funcionando correctamente y, además, ayudó a la correcta implementación del código. Al ver el verdadero resultado a mano, se podía encaminar mejor el problema y fue más fácil detectar pequeños fallos.

# Metodología

Esta sección se dedica a la descripción del método implementado en el trabajo. Esta parte es la correspondiente a lo realmente desarrollado en el trabajo, y se puede emplear pseudocódigo (nunca código), esquemas, tablas, etc.

**Función Nueva Heurística Diferente**

**Entrada**:

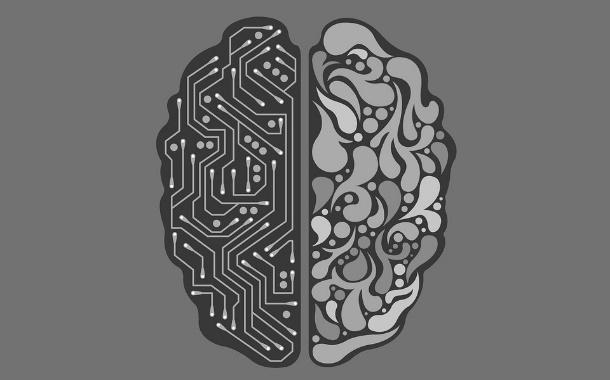
* Estado actual
* Estado objetivo
* Lista de Operadores
* Booleano Heurística

**Salidas**:

* Número entero que denomina la heurística

**Algoritmo**:

1. Si estado actual satisfice objetivo
   1. Devolver lista vacía, es decir, 0
2. Si estado actual no satisface objetivo y no hay ninguna acción que de lugar o satisfaga el objetivo
   1. Devolver lista con todos los operadores, es decir, número total de operadores
3. Sino
   1. Realizar la lista concatenada de la acción que satisface el objetivo y llamada recursiva siendo ahora el nuevo objetivo las precondiciones positivas de la acción que satisface el objetivo.
4. Devolver la lista, es decir, el número de operadores, el cual será la heurística en el nodo recorrido.



1. Ejemplo de un pie de figura. Imágen con derechos Creative Commons

A continuación, un ejemplo de uso de listas numeradas:

### Trabajos con dos alumnos: Poner nombre y apellidos completos de cada uno, y correos electrónicos de contacto (a ser posible de la Universidad de Sevilla). El orden de los alumnos se fijará por orden alfabético según los apellidos.

### Trabajo con un autor: Cambiar la cabecera de la siguiente manera

#### Una sola columna: Sólo se debe especificar un alumno.

#### Información a añadir: La misma que la especificada en el punto 1).

Las figuras se deben mencionar en el texto, como la Fig. 1. También se puede añadir ecuaciones, como la ecuación (1). Un ejemplo de pseudocódigo se puede observar en Pseudocódigo 1.

*a**b* 

# Resultados

En esta sección se detallará tanto los experimentos realizados como los resultados conseguidos:

* Los experimentos realizados, indicando razonadamente la configuración empleada, qué se quiere determinar, y como se ha medido.
* Los resultados obtenidos en cada experimento, explicando en cada caso lo que se ha conseguido.
* Análisis de los resultados, haciendo comparativas y obteniendo conclusiones.

Se Puede hacer uso de tablas, como el ejemplo de la Table 1.

Table 1. Ejemplo de tabla

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

# Conclusiones

Finalmente, se dedica la última sección para indicar las conclusiones obtenidas del trabajo. Se puede dedicar un párrafo para realizar un resumen sucinto del trabajo, con los experimentos y resultados. Seguidamente, uno o dos párrafos con conclusiones. Se suele dedicar un párrafo final con ideas de mejora y trabajo futuro.

##### Referencias

[1] <https://www.iberdrola.com/innovacion/que-es-inteligencia-artificial>

[2] Página web del curso IA de Ingeniería del Software. <https://www.cs.us.es/docencia/aulavirtual/mod/folder/view.php?id=2583>

[3] Automated Planning: Theory and Practice: Chapter 9, Heuristics in Planning

[4] <https://dspace.ups.edu.ec/bitstream/123456789/7543/1/UPS-CT004474.pdf>

[5] <http://idelab.uva.es/algoritmo>

1. S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed, Pearson, 2010.
2. Y. LeCun, Y. Bengio, G. Hinton. “Deep Learning”, Nature, vol. 521, 2015, pp. 436-444.
3. Página web del curso IA de Ingeniería del Software. <https://www.cs.us.es/cursos/iais>. Consultada el 24/03/2018.

Pseudocódigo 1. Algoritmo de Mergesort