Resolución de Puzzles Deslizantes

Lara Rojo Celemín
Ingeniería de Telecomunicación
Universidad Carlos III de Madrid
100072784@alumnos.uc3m.es

Iván Suárez Morejudo
Ingeniería de Telecomunicación
Universidad Carlos III de Madrid
100072643@alumnos.uc3m.es

ABSTRACT

En 1878, Sam Loyd daba un premio de $1.000 a quien fuera capaz de resolver su famoso 14-15 Puzzle. Se trataba de un puzzle deslizante de 16 piezas que no tenía solución, lo que hizo que muchas personas se volvieran locas tratando de resolverlo.

En este trabajo tratamos de presentar diferentes algoritmos para resolver el 15-Puzzle desde el punto de vista de la Inteligencia Artificial. Para ello, utilizamos el algoritmo de búsqueda en árboles A*, del que daremos una visión general que luego particularizaremos a nuestro caso del 15-Puzzle.

Aunque en este trabajo sólo nos hemos centrado en resolver el rompecabezas de 16 piezas, los algoritmos explicados pueden utilizarse para resolver cualquier puzzle deslizante *n x n* con solución.

Términos Generales

Puzzles, Algoritmos, permutaciones pares, heurística, Inteligencia Artificial, resolución humana.

Keywords

Puzzles deslizantes, 15-puzzle, Sam Loyd, Algoritmo A*, herística, Turing.

1. INTRODUCTION

El objetivo de este trabajo es presentar diferentes algoritmos para la resolución de puzzles deslizantes. Aunque sólo nos hemos centrado en el 15-puzzle, los algoritmos aquí presentados pueden ser aplicados a puzzles de cualquier tamaño, siempre y cuando estos tengan solución.

Sin embargo, no sólo nos centraremos en cómo resolverlos de forma manual sino también explicaremos como implementar dichos algoritmos en ordenadores. Para ello, utilizaremos búsquedas en árboles aplicando el *algoritmo de búsqueda A**, el cual explicaremos y particularizaremos.

El primer algoritmo que presentaremos es el algoritmo propuesto por Richard Hayes [4] en el cual el n-puzzle se irá reduciendo a (n – 1)puzzle resolviendo la primera fila y la primera columna.

En el segundo, [5] propone colocar en orden las casillas de las primeras (n – 2) filas. Finalmente, las piezas de estas dos últimas filas se colocarán por columnas. Este método, como veremos más adelante, será el más eficiente.

Finalmente, daremos una visión desde el punto de vista de la psicología sobre las habilidades obtenidas resolviendo este tipo de rompecabezas así como las partes de nuestro cerebro que se ven afectadas. Además explicaremos las diferencias de nuestra mente con respecto a las máquinas y las variables que influyen en la forma de resolver problemas, como es el caso de la ansiedad.

2. EL PUZZLE

2.1 Definición

El *(n^2 - 1)puzzle* consiste en un tablero de *(n x n)* con n casillas, numeradas del 1 al *(n^2 - 1)* , ordenadas de manera aleatoria. La casilla n^2 se encuentra vacía para poder realizar los movimientos necesarios, de forma horizontal o vertical, para poder cambiar el orden de las casillas. Recibe multitud de nombres según el número de fichas *8-puzzle* para la versión *3x3*, *15-puzzle* para la versión de *4x4*, u otros nombres menos comunes como *Puzzle Gem, Puzzle Boss o Cuadrado Místico*. El objeto del rompecabezas es ordenar las fichas utilizando el espacio vacío para desplazarlas.

Figura 1: 15-Puzzle resuelto.

Es un problema clásico de modelación de algoritmos usando heurística, de modo que estos sean eficientes con buenos tiempos de ejecución y usando óptimas soluciones. El uso de la heurística común en este problema conlleva el recuento del número de fichas que se encuentran fuera de lugar, y encontrar la suma de las distancias Manhattan desde su posición actual hasta su posición correcta.

2.2 Historia

No se sabe muy bien a quien atribuir la invención de este rompecabezas ya que la historia es confusa. Los libros, en un principio, la atribuyen a Sam Loyd debido a que en 1878 *"llevara al mundo a la locura"* por un premio de $1.000 a quien fuera capaz de resolver el famoso puzzle de 14-15, el cual no tenía solución debido que se necesitaba un número impar de movimientos y, como veremos más adelante, sólo un número de movimientos par tiene solución.

El puzzle se extendió rápidamente por todas las ciudades de Estados Unidos, Canadá incluso llegó a Europa.

En realidad, el puzzle no fue inventado por Sam Loyd, a pesar que él lo defendiera durante años, sino que era una variación del rompecabezas creado por Palmer Noyes Chapman con los bloques 14-15 invertidos.

Figura 2: Ilustración de Sam Loyd sobre su puzle.

Palmer Noyes Chapman, en Agosto de 1880, presentó ante la oficina de patentes el registro del 15-puzzle. Sin embargo, su solicitud le fue denegada ya que en Febrero de 1878 se había registrado otra patente de un puzzle similar llamado "Puzzle de bloques" por Ernest U. Kinsey.

Previamente, su hijo lo distribuyó en 1879 por diferentes ciudades de Estados Unidos. Por suerte, uno de dichos rompecabezas fue a parar a la Escuela Americana de Sordos de Hartford, quienes lo comenzaron a fabricar en su taller de carpintería para venderlo en las calles de Hartford y Boston. Matthias Rice, propietario de un negocio de carpintería de lujo en Boston, se interesó en la fabricación del mismo y convenció a Yankee Notions (Distribuidor de bienes de lujo) para su comercialización con el nombre de Puzzle Gem.

Casi un siglo después, Ernö Rubik se interesó por el rompecabezas buscando una forma de que funcionara sin el espacio vacío que permitía mover las piezas. Finalmente, su trabajó desembocó en la creación del famoso *Cubo de Rubik.*

3. ALGORITMOS

Para resolver el 15-puzzle podemos aplicar distintos algoritmos que nos lleven a la solución. Como es lógico, unos algoritmos son más eficientes que otros. A continuación se presentan dos algoritmos pero en ambos, la condición necesaria para que tengan solución es que el número de permutaciones sea par.

3.1 Permutaciones pares

Como hemos dicho anteriormente, el 15-puzzle sólo tiene solución si el número de movimientos para resolverlo es par. El número de posibles estados iniciales es n!, siendo n el número de fichas. Por tanto, en nuestro caso, tendremos más de 130.000 millones de posibles estados iniciales. Sin embargo, sólo la mitad de esas combinaciones tiene solución.

Para poder saber la paridad del estado inicial se define el concepto de *Inversión*, es decir, la violación del orden de la posición de acuerdo a su valor. En otras palabras, la inversión de una ficha será la suma del número de fichas que se encuentran en una posición superior a dicha ficha y que deberían estar situadas en una posición inferior. La inversión total será la suma de las inversiones individuales. Si este número es par, el puzzle tendrá solución. En caso contrario, no habrá solución.

Por ejemplo, en el puzzle propuesto por Loyd, la inversión total será uno y, como este número es impar, el número de permutaciones será impar y, por tanto, el puzzle no tiene solución.

Una vez que se conoce la paridad del puzzle, podremos aplicar algún tipo de algoritmo para resolverlo. En nuestro caso, presentamos dos posibles soluciones.

3.2 Algoritmo I

Este algoritmo fue propuesto por Parberry en 1997 para resolver puzles deslizantes *n x n*. Su método se basa en reducir el *n-puzzle* en un *(n-1)-puzzle* resolviendo la primera fila y la primera columna de cada puzzle, quedando como resultado un *(n-1)-puzzle*. Sin embargo, una vez que se llega a un *3x3-puzzle*, este método deja de funcionar ya que no tenemos libertad de movimiento para no afectar a las piezas colindantes y se tiene que resolver usando *"la fuerza bruta"*.

Más tarde, en 2001, Richard Hayes, utilizando el algoritmo de Parberry, consigue evitar el problema de resolver el puzzle usando la fuerza bruta.

El algoritmo se desarrolla en tres pasos:

1. Colocar las piezas de la primera fila
2. Colocar las piezas de la primera columna.
3. Volver al primer paso con el (n-1)puzzle.

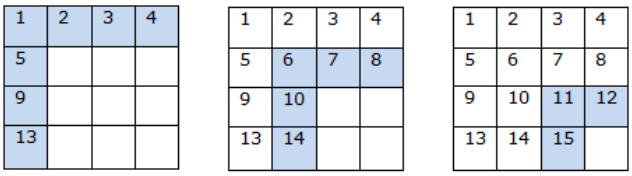

Figura 3: Reducción del 15-puzzle a 3-puzzle.

Para colocar tanto las piezas de la primera fila como de la primera columna de manera eficiente, se toma como referencia las diagonales de la posición en la que debe ir la ficha en cuestión. En función de la posición original de la ficha, utilizaremos una diagonal u otra.

Si la ficha se encuentra a la izquierda de la columna en la que debería ir o en la misma columna, utilizaremos la diagonal de la figura 4a. En cambio, si la ficha se sitúa a la derecha, utilizaremos una de las diagonales descritas en las figuras 4b y 4c.

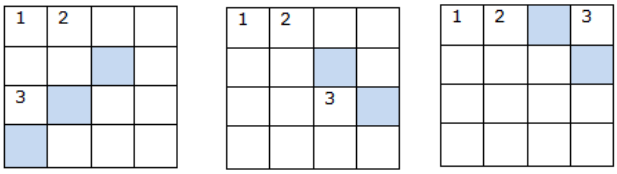

Figura 4: Descripción de las diagonales.

No obstante, las dos últimas fichas de la fila serán colocadas de manera especial. En este caso, se coloca en la posición 1 x (n-1) la ficha (n-2) y, en la posición 2 x (n-2) la ficha (n-2) siguiendo el mismo procedimiento explicado con anterioridad. Por último, se desplaza la ficha(n – 2) a la posición 1 x (n – 2) y la (n – 1) a su posición.

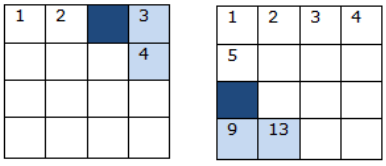

Figura 5: Colocación de las últimas piezas.

Para las columnas, se sigue un procedimiento similar: se colocan las primeras piezas y cuando queden sólo dos, se pondrá la penúltima pieza en la última posición y la última pieza en la columna adyacente a la última posición.

Sin embargo, hay algunas combinaciones que no pueden ser resueltas aplicando este algoritmo. Para ello, buscamos una posible combinación tal que se pueda resolver a través del algoritmo previamente descrito.

Las únicas combinaciones que no tienen solución ocurren cuando la pieza (n- 1) de una fila se sitúa en la posición (n – 2) o cuando la pieza (n – 1) de una columna se encuentra en ((n – 2), 0). Por ello, lo único que debemos hacer es mover dicha pieza de esa posición.

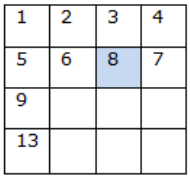

Figura 6: Caso especial.

3.3 Algoritmo II

A partir de la información obtenida en [4], podemos resolver el 15-Puzzle ordenando la primera y segunda fila de manera consecutiva, para luego terminar de ordenar las dos últimas filas por columnas. De esta manera se consigue llegar al estado final del puzzle en un promedio de 80 movimientos, una cifra bastante eficiente teniendo en cuenta el número de movimientos posibles. La implementación de dicho método se basa en el *Algoritmo de búsqueda A**.

*3.3.1 Algoritmo de búsqueda A**

El algoritmo de búsqueda A* es un algoritmo computacional clasificado dentro de la búsqueda por grafos, para encontrar el camino de menor coste entre el nodo origen y destino. Está motivado debido a que en algoritmos de búsqueda en grafos informados, como el algoritmo voraz, estos siguen un camino basado únicamente en la función heurística. Dicha función no nos indica el coste real de desplazarse de un nodo a otro, sino una aproximación que puede no ser óptima, llevando a realizar movimientos extra que terminan derivando en un mayor coste para alcanzar la solución.

Por ello un algoritmo óptimo de búsqueda informada deberá tener en cuenta el factor del valor heurístico de los nodos, y el coste real del recorrido. Por ello el algoritmo A* usa la siguiente función de evaluación $\boldsymbol{f(n) = g(n) + h'(n)}$, siendo el primer término g(n) el coste real del camino recorrido para llegar al nodo n, y h'(n) el valor heurístico del nodo a evaluar desde el actual n hasta el final.

Además A* necesita de dos estructuras de datos auxiliares, para mantener un conjunto de soluciones parciales almacenadas:

- *Abiertos*: se trata de una cola de prioridad ordenada según el valor f(n) de cada nodo.
- *Cerrados*: información de los nodos ya visitados.

En cada iteración del algoritmo se realiza una consulta de la estructura de *Abiertos*, se consulta el primer nodo de la lista ordenada, si no se trata de un nodo objetivo, se calcula la f(n) de

todos sus hijos, se insertan en la cola de *Abiertos*, y finalmente el nodo evaluado se traspasa a la estructura de *Cerrados*.

Se trata de un algoritmo de búsquedas de tipo primero en anchura para g(n) con primero en profundidad para h'(n). Esto quiere decir que:

- Para g(n) se establece un nodo raíz, y se exploran y evalúan todos los vecinos de este nodo, y a continuación para cada uno de los vecinos se exploraran sus correspondientes vecinos adyacentes, y se realiza esto hasta que hayamos explorado todo el árbol.
- Para h'(n) el modo de exploración es diferente, en la búsqueda en profundidad partimos de un nodo que vamos expandiendo de manera recurrente para un camino determinado, en el momento que ese camino no se puede propagar más, volvemos atrás y comenzamos de nuevo a expandir el nodo vecino del cual partimos y que ya ha sido procesado. Es un algoritmo que nos permite recorrer todo el árbol de manera ordenada pero no uniforme.

La complejidad y carga computacional de este algoritmo está dispuesta por la calidad heurística usada en el problema. Por ello para una mala heurística la complejidad será exponencial, y para una heurística óptima tendremos una complejidad lineal y por ello un tiempo de ejecución del mismo modo, siendo necesario por ello, que con una heurística optima se verifique que:

$$h'(x) <= g(y) - g(x) + h'(y)$$

,donde h'(y) es un estimador de h(x) que indica la distancia a la solución.

Las propiedades del algoritmo son las siguientes:

- Es un algoritmo completo, en caso de existir solución la encontrará.
- El algoritmo no desarrolla un camino por interacción, formula varios y selecciona lo más competentes.
- Para garantizar que se trata de un algoritmo óptimo, la función h(n) deberá ser válida, sin exagerar el valor real de encontrar la solución.
- SI h'(n) realiza una estimación exacta de h(n), el algoritmo converge rápidamente a la solución.
- Si h'(x) = 0, la búsqueda es controlada por la función g(x).
- Si h'(x) = g(x) = 0 se trata de una búsqueda aleatoria.
- Si h'(x) = 0 y g(x) = 1, se trata de una búsqueda que se desarrolla primero en anchura.
- Si h(x) no se sobrestima por h'(x), se encuentra un camino optimo pero se han buscado rutas alternativas que han conllevado un sobre cálculo desaprovechado.
- En caso contrario si h(x) se sobrestima por h'(x), no se asegura que se avance por el camino de menor coste.
- Si para todos los nodos n del grafo se verifica que g(n) = 0 se trata de una búsqueda voraz.
- Si para todos los nodos n del grafo se verifica que h(n) = 0, se trata de una búsqueda de coste uniforme no informada.

Pero en este algoritmo no todo son ventajas, existe un gran problema a la hora de ejecutar un algoritmo basado en A*, que es la gran cantidad necesaria de memoria. Esto es debido a que se debe almacenar todos los posibles siguientes nodos de cada estado, derivando por ello en una cantidad de memoria exponencial respecto al tamaño y complejidad del problema. Como alternativas a este problema se presentan variaciones de este algoritmo como pueden ser RTA*, IDA* o SMA*.

7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

Figura 7: Ejemplo de aplicación del algoritmo A*.

A continuación explicamos el funcionamiento del algoritmo mediante un ejemplo. En dicho ejemplo buscamos el camino a recorrer más corto para ir desde el punto verde al rojo, esquivando el muro azul el cual no podemos atravesar. Como observamos el área de búsqueda se ha dividido en una simple rejilla, simplificando así el problema a una matriz bidimensional, representando así cada cuadrado de la rejilla como un elemento de la rejilla, y pudiendo así almacenar sus propiedades y valores, como si es posible atravesarlo. A partir de ahora a dichos cuadrados los llamaremos nodos.

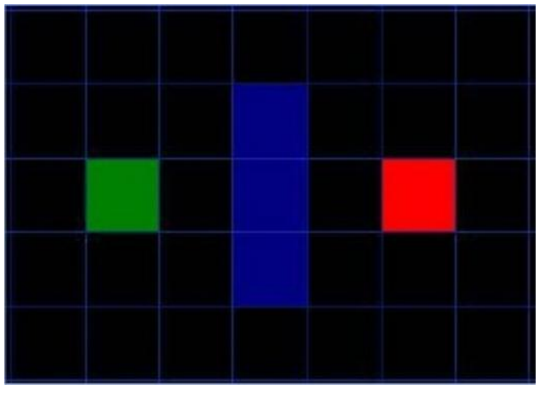

Figura 8: Área de búsqueda en rejilla. En verde, punto de inicio. En rojo, punto final.

Comienza la búsqueda:

1. Comenzamos por el nodo de inicio (verde), añadimos este a una lista abierta, la cual está formada por los nodos que pueden formar parte del camino a seguir, pero que aún debemos comprobar.

2. Revisamos todos los nodos alcanzables y colindantes al nodo de inicio, ignorando los cuadrados intransitables como puede ser el muro. Los añadimos a la lista abierta, marcando en cada uno de ellos una referencia al nodo de inicio como su nodo padre. Bordeamos los nodos de la lista abierta en color verde, y marcamos la referencia al nodo padre con una flecha gris.

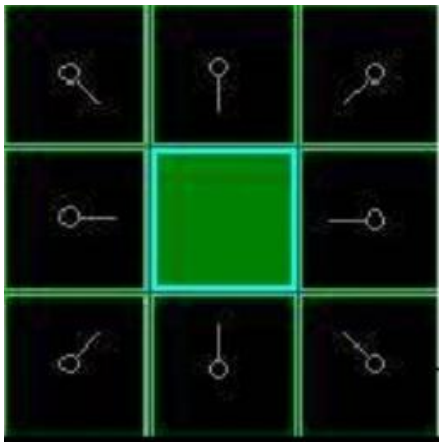

Figura 9: Nodo de inicio y sus nodos alcanzables.

3. Traspasamos el nodo inicio a la lista cerrada, ya que no es necesario para nada de momento. El borde de su color en la imagen será azul, debido a que se encuentra en la lista cerrada.

4. Ahora deberemos seleccionar el nodo con el valor de F (F = G+H) más bajo. Donde G es el valor de movimiento del nodo inicio a otro nodo de la rejilla siguiendo el camino necesario, y siendo F el valor del movimiento estimado para ir desde ese nodo hasta el nodo final, donde este último valor no es más que una hipótesis ya que no conocemos aún la distancia exacta.

    a. En este ejemplo al valor de G le daremos un valor de 10 a los movimientos ortogonales, y de 14 a los movimientos diagonales. Para el cálculo del valor G del nodo, sumaremos al valor G de su padre el valor del tipo de movimiento que realicemos.

    b. En el caso del valor H este se puede estimar de muchas maneras diferentes. En este caso utilizaremos una aproximación mediante la distancia Manhattan, donde solo tendremos en cuenta el número de cuadrados horizontales y verticales (solo existe movimiento ortogonal) que debemos desplazarnos de manera directa (ignorando y no teniendo en cuenta los obstáculos que pudieran existir en el recorrido) para llegar desde el nodo inicio al nodo de destino. Como observamos se trata de una aproximación del recorrido a seguir, de ahí que se trate de un valor heurístico.

En la Figura [9] podemos observar los valores obtenidos, donde el valor G está representado en la esquina inferior izquierda, H se encuentra en la esquina inferior derecha, y el valor F de resultante de la suma de G y H se encuentra en la esquina superior izquierda. También vemos que hemos elegido el nodo que colinda a la derecha con el nodo inicio (marcado con el borde azul), ya que se trata del nodo con menor valor de F, en este caso 40.

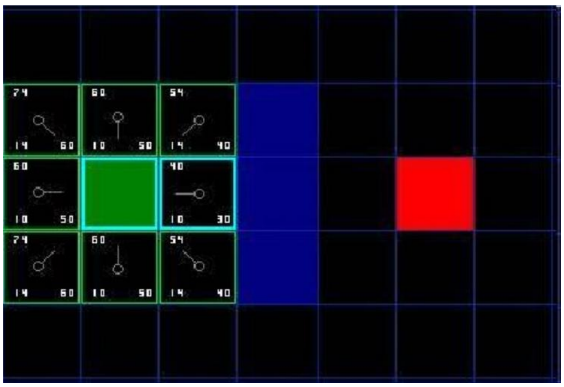

Figura 9: Valores de las funciones.

5. Traspasamos el nodo seleccionado de la lista abierta a la cerrada, ya que no vamos a operar más con él, y cambiamos por ello el color de su borde de verde a azul.

6. Verificamos que los nodos contiguos que sean accesibles y no se encuentren en la lista cerrada, y añadimos a la lista abierta a aquellos que no pertenecen aún a ella, marcando como nodo padre al seleccionado anteriormente.

7. Comprobamos para los nodos contiguos que ya se encontraban en la lista abierta, si su valor G es inferior al nuevo valor de G que recalcularíamos con el nodo actualmente seleccionado. Si el valor G del nuevo camino es menor, seleccionamos este nuevo nodo, y cambiamos el padre de los nodos contiguos por este otro, recalculando de estos nodos su nuevo valor de G y F.

En nuestro caso como observamos, el valor obtenido de G para el nodo inferior derecho moviéndonos en diagonal desde el nodo inicio tiene un valor de 14, frente al valor G igual a 20 que tendría si el camino recorrido por dicho nodo, proviniera del nodo superior, que previamente provenía por su izquierda del nodo inicial. Por ello seleccionamos este nuevo nodo (marcándolo como azul), desechamos el anterior. En el caso de existir un caso con dos posibilidades seleccionamos aquella con valor F menor.

8. Marcamos y apuntamos el nuevo nodo seleccionado como padre para los nodos contiguos. Comprobamos los nodos contiguos y calculamos sus valores de G, H y F. Podemos observar el proceso en la Figura [10].

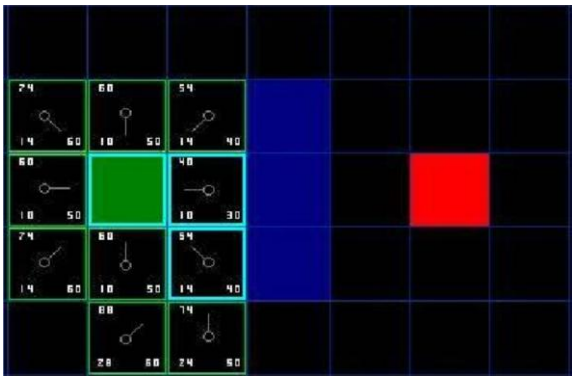

Figura 10: Marcamos y calculamos las funciones de las casillas colindantes al nuevo nodo.

9. Comprobamos si el nodo situado a la izquierda del ahora seleccionado tiene un coste G menor a través del nodo actual, frente a llegar desde el nodo inicial. Como no es así lo desechamos.

10. Repetimos de manera recursiva este proceso, hasta que finalmente añadimos el nodo destino a la lista abierta. Una vez llegado a ese estado obtendremos algo similar a lo que podemos apreciar en la Figura [11].

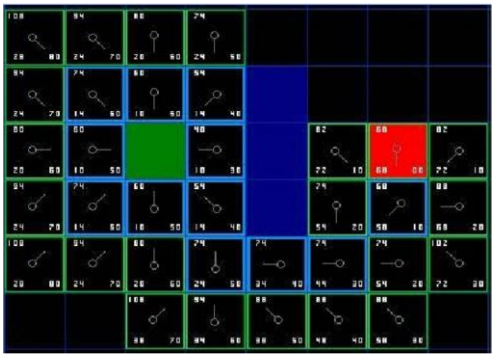

Figura 11: Resultados tras aplicar el algoritmo.

11. El camino buscado que une el nodo inicial con el final, es aquel el cual apuntan las flechas partiendo desde el nodo final, recorriéndolas hasta el nodo inicial. En la Figura [12], observamos el camino obtenido para esta interacción.

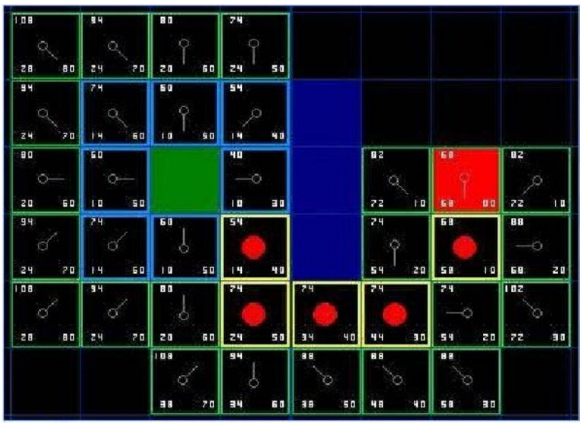

Figura 12: Camino obtenido para llegar del nodo origen al destino aplicando este algoritmo de búsqueda.

Aplicando este algoritmo para el caso del 15 Puzzle, nuestros nodos serán las posibles combinaciones que conseguimos desplazando las fichas a través de la casilla vacía. A través del siguiente ejemplo explicaremos su funcionamiento.

Partimos del siguiente estado inicial, Figura [13]. Primero comprobaremos el número de permutaciones, que en este caso es par (4). A continuación, creamos las dos listas o estructuras de datos: Abierta y Cerrada. En la abierta, insertamos nuestro estado inicial.

1	2	3	4
5	6	7	8
9	10	12	15
13	14	11	

Figura 13: Estado Inicial de nuestro Puzzle.

Ahora tenemos que analizar los nodos alcanzables a nuestro nodo origen, es decir, definimos los posibles movimientos que podemos realizar. En este caso, sólo tenemos dos posibilidades (Figura [14])

12	15
	11

12	
11	15

Figura 14: Posibles movimientos.

Hay que calcular el valor de la función *f(n)*. Dado que estamos realizando un ejemplo sencillo y sólo hacemos movimientos ortogonales (horizontal o vertical), tomaremos como función g(n) el número de movimientos realizados hasta llegar a ese estado. En cuanto a la función heurística *h(n)*, utilizaremos la suma del número de casillas que se encuentran en una posición superior con respecto a la casilla analizada, sin tener en cuenta la casilla vacía. Para ello, la forma más sencilla es colocar nuestro tablero en forma de vector:

Estado inicialA: 1 2 3 4 5 6 7 8 9 10 12 15 13 14 – 11

Estado final: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 –

$g_A(n) = 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 + 3 + 1+1 = 6$

Estado inicialB: 1 2 3 4 5 6 7 8 9 10 12 15 13 14 – 11

Estado final: 1 2 3 4 5 6 7 8 9 10 11 – 13 14 11 15

$g_B(n) = 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 + 1+1 = 3$

Por tanto, obtendremos los siguientes valores para la función *f(n):*

$$f_A(n) = g_A(n) + h_A(n) = 1 + 6 = 7$$

$$f_B(n) = g_B(n) + h_B(n) = 1 + 3 = 4$$

A continuación pasamos nuestro estado origen a la lista Cerrada y añadimos en la lista Abierta nuestros nuevos estados ordenados en orden creciente de acuerdo al valor obtenido de la función.

Nuestro nodo origen será ahora el estado de la figura [14]b. Ahora sólo obtendremos un nuevo estado: Figura [15]

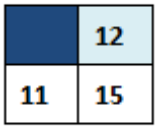

Figura 15: Nuevo estado inicial.

La función *f(n)* será ahora 5, así que será nuestro siguiente nodo origen.

Aplicando dos veces más este algoritmo, llegamos a la solución del puzzle en tan sólo 4 movimientos.

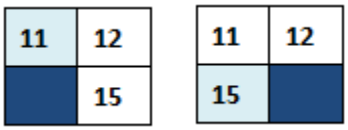

Figura [16]: Últimos estados iniciales.

Finalmente, en la Figura [17] se muestra el árbol resultante tras aplicar el algoritmo:

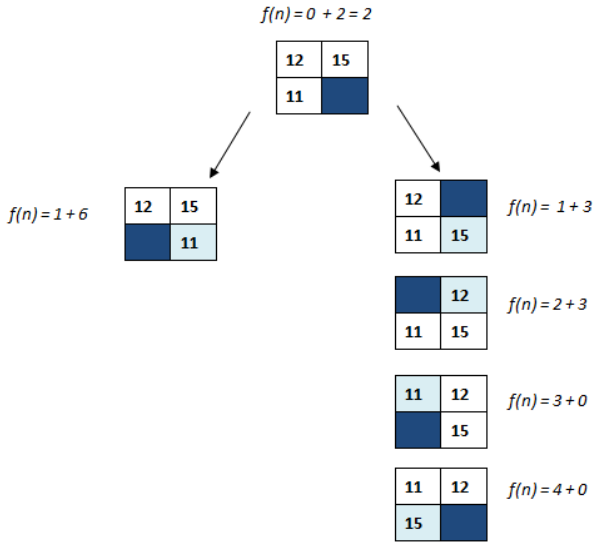

Figura [17]: Árbol resultante.

4. ENFOQUE PSICOLÓGICO PARA EL PLANTEAMIENTO DE ALGORITMOS

En las últimas décadas, debido al desarrollo tecnológico, ha tenido una especial importancia el debate sobre si la mente humana puede compararse con el ordenador. Para ello, se han realizado estudios que abordaban la resolución de problemas tanto en humanos como en ordenadores. En Psicología han destacado dos autores que han tratado este tema: Turing afirmó que la mente humana se puede comparar con el ordenador debido a que ambos pueden razonar lógicamente, mientras que Searle se opuso a esta afirmación defendiendo que, si bien es cierto que ambos pueden manejar símbolos, sólo la mente humana es capaz de atribuir significado a los símbolos que maneja, por ello, según este autor la mente no es un programa informático. No obstante, para resolver determinadas tareas no es necesaria la intervención del significado, por ejemplo, en la solución de este puzle basta con manejar los símbolos adecuadamente.

Turing es considerado uno de los padres de la ciencia de la computación ya que formalizó los conceptos de algoritmo y computación mediante su famosa máquina de Turing. Además, es uno de los precursores de la Inteligencia Artificial ya que consideraba que las máquinas podrían llegar a pensar. Lo cual enunció a través de la prueba que lleva su nombre, en la que pretendía comprobar de manera estándar si una máquina puede apuntar a ser sensible o sintiente.

Con respecto a la anatomía cerebral, cabe destacar que el cerebro se divide en dos hemisferios y para resolver este puzle se requieren ambos. Por un lado, el hemisferio izquierdo se relaciona con el razonamiento y la habilidad numérica, puesto que es analítico, racional, secuencial, organizado, matemático, simbólico, etc. Mientras que, por otro lado, el hemisferio derecho se corresponde con la habilidad visoespacial. Las habilidades

mencionadas son necesarias para resolver el puzle y estos datos se pueden apoyar mediante técnicas cerebrales como el EEG que muestran las áreas cerebrales que se activan ante esta tarea. Además, las alteraciones en determinadas zonas del cerebro suelen ir acompañadas de una dificultad a la hora de resolver este tipo de problemas.

Esta especialización cerebral se produce por la necesidad que tiene el cerebro de ahorrar energía, lo que nos da una similitud con la programación que nos conlleva siempre a diseñar programas especializados para hacer un uso eficiente de los recursos disponibles y/o limitados de la máquina para su ejecución.

En cuanto a la solución de problemas, los seres humanos aplican una serie de estrategias según el tipo de problema. Para resolver este puzle la estrategia óptima es la división del problema en subproblemas o submetas. Esta estrategia se basa en la idea de "divide y vencerás", por lo que se trata de reducir un problema amplio en varios más pequeños que acerquen a la persona a la meta.

Para la Gestalt, la solución de problemas es algo más que aplicar de forma mecánica la experiencia, es realizar una reestructuración perceptiva. Este enfoque diferencia entre pensamiento productivo y reproductivo: el pensamiento productivo es la producción de una solución nueva a partir de una organización creativa del problema (se usa cuando se ha comprendido la estructura del problema), por ello, para la Gestalt éste es el verdadero pensamiento; por otro lado, el pensamiento reproductivo consiste en aplicar en situaciones similares la misma solución. Para la resolución del puzle se requiere el pensamiento productivo.

Por otro lado, según Greeno, los problemas se pueden dividir en tres clases principales: Problemas de deducción de la estructura, Problemas de organización y Problemas de transformación. Este puzle es un problema de transformación, puesto que tiene todos los elementos del estado inicial y, a medida que se va avanzando y se van aplicando operadores, se va transformando el problema hasta llegar a la meta.

Sumado a ello, cabe mencionar los cuatro pasos necesarios para resolver un problema propuestos por Polya (1945). En primer lugar, es necesaria la comprensión del problema, esto supone no sólo descodificar el lenguaje del enunciado, sino también adoptar una actitud de solución de la tarea e identificar el problema. Algunos ejemplos de preguntas que se pueden proponer en esta fase son: ¿cuál es la meta?, ¿cuáles son los datos?, ¿qué condiciones? y ¿es suficiente la información?. En segundo lugar se elabora un plan. Las preguntas características de este paso son: ¿conozco algún problema relacionado con éste?, ¿es necesario dividir el problema? y ¿es necesario resolver alguna parte antes que otra?. En tercer lugar, se ejecuta el plan, se comprueba cada paso y se resuelve el problema. Algunas de las preguntas de esta paso son: ¿es necesario este paso?, ¿es correcto?. Por último, el cuarto paso contempla la visión retrospectiva o revisión de la tarea, esto es, verificar el resultado y el razonamiento. Algunos ejemplos de preguntas son: ¿se puede obtener el mismo resultado por algún otro método?. Estos cuatro pasos son descriptivos y normativos y, como puede observarse, mantienen relación con el proceso que lleva a cabo un ordenador a la hora de resolver este tipo de problemas.

No obstante, en determinadas ocasiones la mente humana no sigue estos pasos ininterrumpidamente, sino que llega a un punto en el que la persona no se ve capaz de resolver el problema y lo abandona durante un tiempo en el que la mente de forma inconsciente sigue pensando en el problema hasta dar con la solución. Este fenómeno recibe el nombre de *insight* y es uno de los aspectos que diferencia a los seres humanos de los ordenadores. Sumado a ello, los seres humanos presentan otras variables que influyen en la forma de resolver problemas, por ejemplo, la ansiedad, que facilita hasta cierto punto la resolución de la tarea y, pasado ese punto, la dificulta. Por ello, la relación entre rendimiento y activación se representa siguiendo la Ley de Yerkes-Dodson:

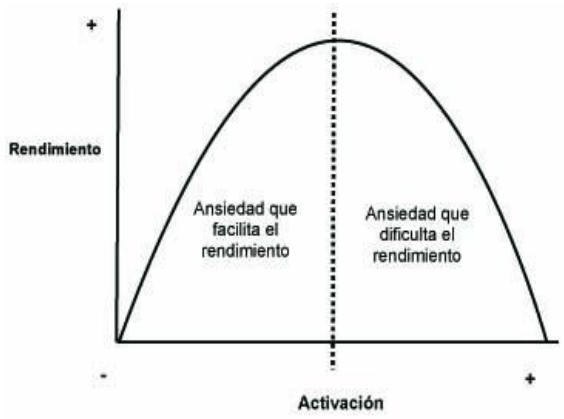

Figura 18: Ley de Yerkes-Dodson.

Otro punto a destacar es el papel que tiene el aprendizaje en la resolución de este tipo de puzle. Por un lado, existen teorías sobre la solución de problemas basadas en habilidades generales, como la Gestalt y el Procesamiento de la Información, mientras que otras teorías defienden que la solución de problemas se basa en habilidades específicas, como la teoría de Expertos y novatos. Para el primer enfoque se puede aprender a resolver este tipo de problemas de forma que el aprendizaje afecte a las habilidades generales. No obstante, según el enfoque de Expertos y novatos el aprendizaje solo se produce en una habilidad específica, puesto que, por ejemplo, aprender a resolver un puzle no mejora la capacidad para resolver problemas de índole social. La eficacia en la solución de un problema no depende de la disposición de estrategias o habilidades generales y transferibles, válidas para cualquier problema, sino de los conocimientos específicos, útiles para solucionar ese problema. El rendimiento experto sería el modelo para la solución eficiente de un problema. Hay dos tipos de expertos: los técnicos y los estratégicos. Los técnicos resuelven un problema de su dominio de forma automática y rutinaria. Los estratégicos resuelven un problema nuevo o poco familiar dentro de su dominio sin contar con procedimientos automáticos, por ello, requieren un control metacognitivo.

Los expertos son mejores que los novatos resolviendo problemas incluso cuando se enfrentan con problemas fuera de su área de pericia porque manejan más recursos cognitivos.

- La eficacia en la solución de problemas no se debe a diferencias en la capacidad cognitiva general, sino a las diferencias en conocimientos específicos.

- El conocimiento procedural: Por un lado, los expertos y los novatos utilizan distintos heurísticos en la solución de los mismos problemas. Además, los expertos controlan mejor la ejecución del problema. Por último, los expertos utilizan un amplio número de algoritmos de forma automática y, además los generalizan a diversos problemas. Los expertos aplican técnicas y no estrategias (uso deliberado e intencional).

Sumado a ello, la teoría de Expertos y novatos destaca el papel del conocimiento previo, lo que nos lleva a una de las teorías más relevantes de la Psicología de la Educación: el Constructivismo. Pese a que la idea de que una persona no es experta en todo sino que sólo en unos dominios, como propone la teoría de Expertos y novatos, es opuesta a la teoría de Piaget, también es cierto que ambas teorías recogen la importancia de la experiencia o conocimiento previo debido a la influencia que tiene a la hora de resolver un problema de este tipo. Por ejemplo, si una persona a aplicado anteriormente una estrategia inadecuada para resolver el puzle, en el futuro será menos probable que vuelva a aplicar la misma estrategia, por lo que dará antes con la estrategia óptima y solucionará la tarea con mayor rapidez.

Por su parte, esta relación entre la conducta y sus consecuencias ya la propuso el Conductismo cuyo fundador fue Watson, destacando el condicionamiento instrumental u operante desarrollado por Skinner, cuya premisa principal es que aquella conducta que vaya seguida de una consecuencia negativa (castigo) tendrá menor probabilidad de repetirse, mientras que la conducta seguida de una consecuencia positiva (refuerzo) probablemente se repita en un futuro.

Referencias

[1] Fifteen Puzzle
http://en.wikipedia.org/wiki/15-puzzle

[2] Slocum, J. Sonneveld D. 2006. *The 15 Puzzle: How it drove the World crazy*. The Slocum puzzle Foundation.

[3] Juego del Puzzle
http://es.scribd.com/doc/41776980/Juego-Del-Puzzle-1

[4] Hayes, R. 2001. *The Sam Loyds 15 Puzzle.*

[5] Parberry, I. 1997. *A Real-Time Algorithm for the (n^2-1)-Puzzle*. University of North Texas. Department of Computer Science.

[6] Búsqueda en árboles
http://es.scribd.com/doc/19950923/Busqueda-Heuristica

[7] Algoritmo A*
http://es.wikipedia.org/wiki/Algoritmo_de_b%C3%BAsqueda_A*

[8] A* search algorithm
http://en.wikipedia.org/wiki/A*_search_algorithm

[9] Zorahn. 2010. *Applying Search Algorithms to Turn-Based Games.* http://forum.codecall.net/blogs/zoranh/1323-applying-search-algorithms-turn-based-games.html

[10] Carreteto, M. 2008. *Psicología del Pensamiento: Teoría y prácticas*. Madrid: Alianza

[11] Carretero, M. 2009. *Constructivismo y educación*. Buenos Aires: Paidós.

[12] Haines, D. E. 2003. *Principios de Neurociencia*. Madrid: Elsevier.

[13] Hergenhahn, B. R. 2001. *Introducción a la Historia de la Psicología.* Madrid Paraninfo.

[14] Alan Turing
http://es.wikipedia.org/wiki/Alan_Turing

[15] Weiten, W. 2004. Themes and Variations. Thomson-Wadsworth