Trabajo Fin de Grado

Grado en Ingeniería de Organización Industrial

Resolución del juego Unblock Me utilizando programación matemática

Autor: Manuel Jesús Rivero Rufino

Tutor: José Manuel García Sánchez

Dep. Organización Industrial y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Grado
Grado en Ingeniería de Organización Industrial

Resolución del juego Unblock Me utilizando programación matemática

Autor:

Manuel Jesús Rivero Rufino

Tutor:

José Manuel García Sánchez

Profesor titular

Dep. Organización Industrial y Gestión de Empresas I

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Grado: Resolución del juego Unblock Me utilizando programación matemática

Autor: Manuel Jesús Rivero Rufino
Tutor: José Manuel García Sánchez

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

ÍNDICE

1. INTRODUCCIÓN Y OBJETIVOS DEL TRABAJO

En este trabajo fin de grado se ha realizado un estudio sobre el uso de programación matemática para la resolución de un juego para dispositivos móviles llamado *Unblock Me Free*, mediante el cual se han propuesto dos modelos matemáticos que resuelven los rompecabezas del juego.

El juego consiste en la resolución de un rompecabezas en el cual hay que sacar el bloque de color rojo por una abertura que presenta el panel. Para ello, hay que desplazar el resto de piezas despejando el camino central donde se encuentra el bloque objetivo, intentando resolver el tablero en el menor número de movimientos posibles. En la imagen 1.1 vemos un ejemplo de rompecabezas donde observamos el bloque objetivo de color rojo y los otros de color marrón.

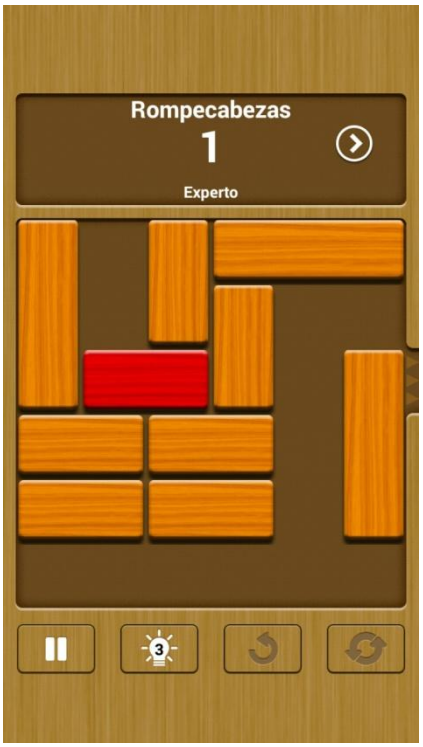

Imagen 1.1: Ejemplo de rompecabezas

En la primera parte del trabajo, que se corresponde con el capítulo 2, se describirá en mayor profundidad el juego en sí, explicando los niveles en los que se divide el juego, así como las diferentes opciones de las que consta.

La segunda parte, capítulo 3, se corresponde con la formulación matemática del juego, en la que se detallará el modelo matemático de cada uno de los anteriormente citados, incluyendo una descripción de cada una de las variables, índices que se emplean y datos que se extraen del juego.

En el capítulo 4 se desarrollará la implementación de los modelos matemáticos en una herramienta de programación lineal, denominada LINGO, mediante la cual obtendremos los resultados de la resolución de cada rompecabezas.

Posteriormente, en el capítulo 5, describiremos los resultados del experimento. Además, se explicarán los puzzles que se han utilizado como datos para resolver cada problema, así como una comparativa de tiempos de resolución, tanto con distintas personas que han solucionado manualmente cada puzzle como con el uso de los modelos.

Por último, en el capítulo 6, se presentarán las conclusiones obtenidas a partir de los resultados conseguidos, así como las posibles utilidades futuras de este estudio. Además, se incluirá la bibliografía utilizada para llevar a cabo el trabajo.

1.1. Objetivos

El objetivo principal de este proyecto es analizar el comportamiento del uso de la programación matemática para resolver los distintos rompecabezas del juego *Unblock Me Free*.

Para alcanzar al objetivo principal, se han desarrollado los siguientes objetivos secundarios:

- Implementación de modelos matemáticos en herramientas de programación lineal: LINGO y Visual Basic.
- Análisis de las diferentes estrategias empleadas para la resolución de los distintos problemas.
- Análisis del comportamiento de los modelos respecto al tamaño del problema.
- Análisis del comportamiento de los modelos empleados respecto al número de fichas que tiene cada puzle.
- Comparación de la resolución computacional con la resolución manual en cuanto al tiempo empleado para llegar a la solución.
- Comparación de la resolución computacional con la resolución manual en cuanto al número de movimientos empleados para llegar a la solución.

2. DESCRIPCIÓN DEL JUEGO

2.1. Introducción y entorno

Unblock Me Free es un juego para dispositivos móviles cuyo objetivo principal es sacar el bloque de color rojo fuera del tablero. Para ello, hay que deslizar el resto de bloques de color marrón para despejar el camino del bloque objetivo en el menor número de movimientos posibles. En la siguiente imagen (imagen 2.1), podemos ver un ejemplo de un panel del juego.

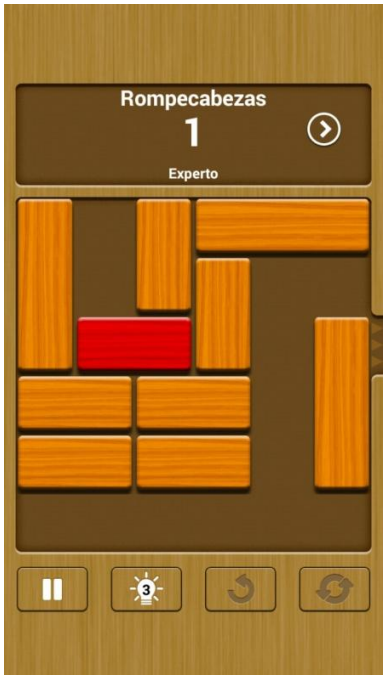

Imagen 2.1: ejemplo de panel

La primera pantalla que encontramos al abrir el juego, como se ejemplifica en la imagen 2.2, nos muestra un menú en el que encontramos las opciones: jugar, rompecabezas, opciones del juego y tienda.

Imagen 2.2: pantalla principal

Tanto en la opción jugar como en rompecabezas aparece una nueva pantalla, en la que tenemos la opción de elegir el modo de juego, que puede ser relax, si solo se quiere jugar sin competir, desafío, si se busca conseguir una puntuación alta o competir contra los valores que por defecto el juego propone para superarlos, o multijugador, si se prefiere competir contra otros jugadores. En la imagen 2.3 podemos observar los distintos modos de juego.

Imagen 2.3: seleccionar modo

En el modo relax, el juego no tiene en cuenta el número de movimientos empleados, mientras que en el modo desafío sí se tiene en cuenta, incluyendo el número de movimientos récord. Así como también se incluyen hasta un máximo de tres estrellas, que se conseguirán según el número de movimientos empleados para despejar el panel. En la imagen 2.4 vemos una comparación entre el rompecabezas 1 en el modo relax y en el modo desafío, en la cual vemos lo descrito anteriormente.

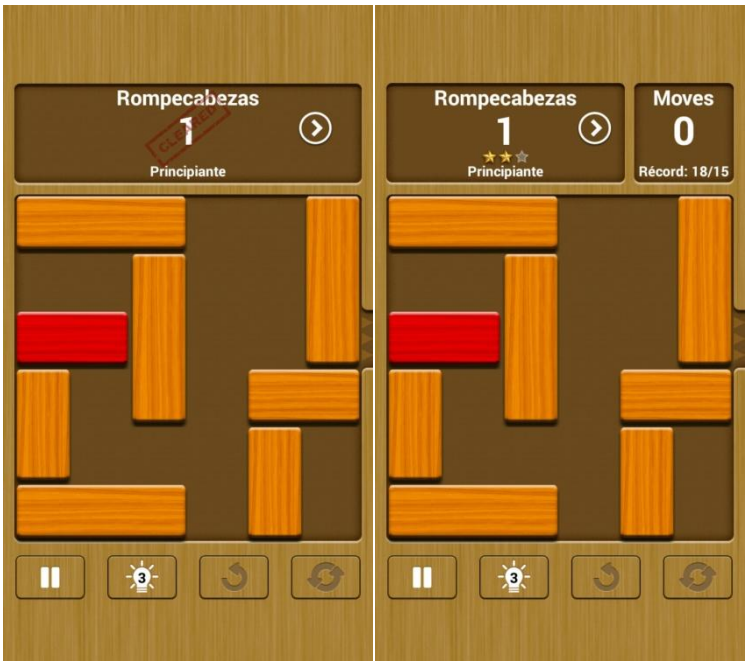

Imagen 2.4: comparación modo relax rompecabezas 1 y modo desafío rompecabezas 1

Dentro de la opción rompecabezas, como se ha mencionado anteriormente, también podemos seleccionar el modo de juego. Los modos de juego siguen siendo los mismos pero esta vez puedes elegir el grado de dificultad, que se divide en principiante, intermedio, avanzado, experto y original gratuito. Una vez seleccionado el nivel al que queremos jugar, podemos elegir diferentes packs dentro del mismo nivel, como por ejemplo, original, bonus, gratuito, entre otros. En la imagen 2.5 podemos observar los distintos subniveles del nivel principiante.

Imagen 2.5: pack principiante

Después de haber elegido el paquete, se despliegan los diferentes paneles que podemos intentar resolver. Por ejemplo, en el nivel principiante, si elegimos el paquete original, tenemos la opción de escoger entre 600 paneles distintos. En la imagen 2.6 podemos ver los primeros 9 puzzles del nivel principiante, dentro del paquete original.

Imagen 2.6: rompecabezas nivel principiante paquete original

2.2. Descripción de los movimientos de un jugador

En este subcapítulo, se va a explicar mediante imágenes, los distintos movimientos que realiza un jugador cualquiera para poder solucionar un puzzle del juego.

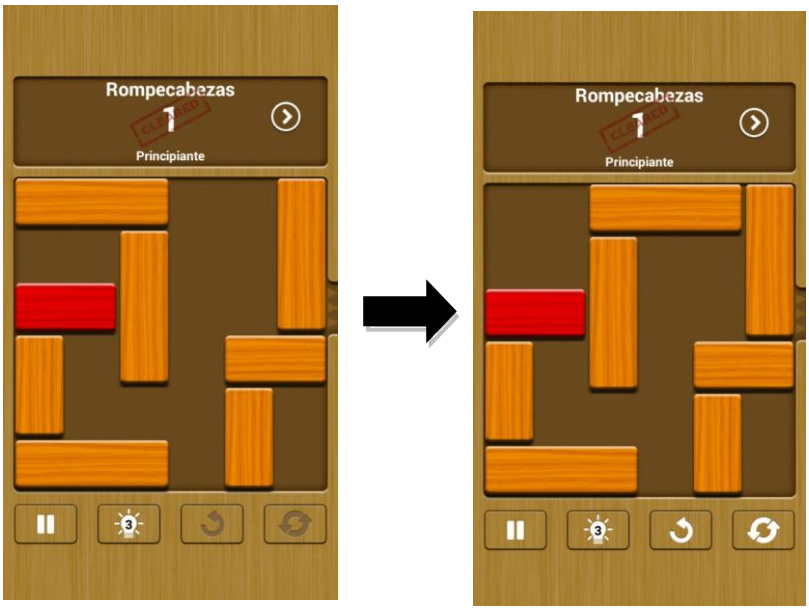

Imagen 2.7: ejemplo de solución de un panel. En la imagen izquierda podemos ver el panel al que se enfrenta el jugador para resolver. En la de la derecha, el jugador ha desplazado la pieza de arriba de tamaño 3 dos posiciones a la derecha, dejando las demás piezas en la posición inicial en la que se encontraban.

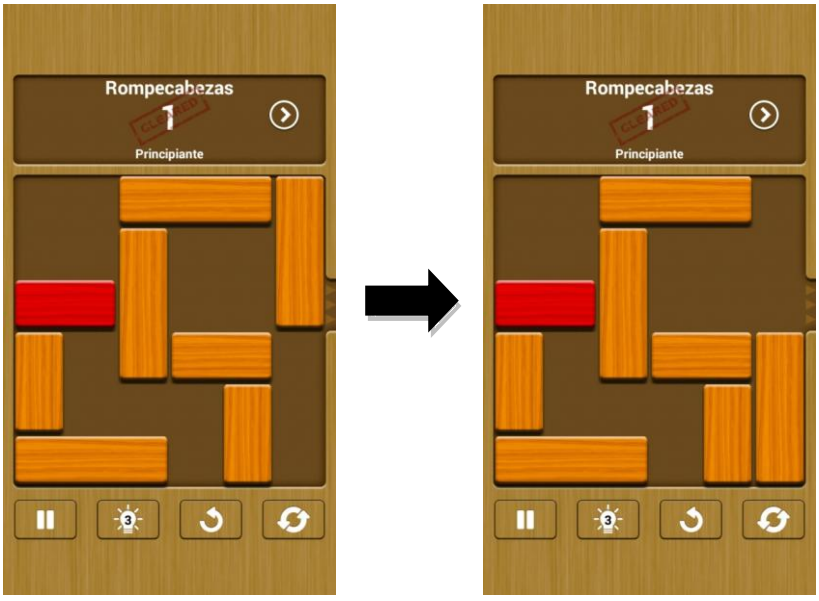

Imagen 2.8: ejemplo de solución de un panel. En la primera imagen observamos que el jugador ha movido la pieza horizontal de tamaño 2 más cercana a la salida (justo debajo de ella) una posición a la izquierda. En la otra imagen, ha bajado la pieza

vertical de tamaño 3, que estaba tapando la salida, hasta la posición más baja posible, liberando así la salida.

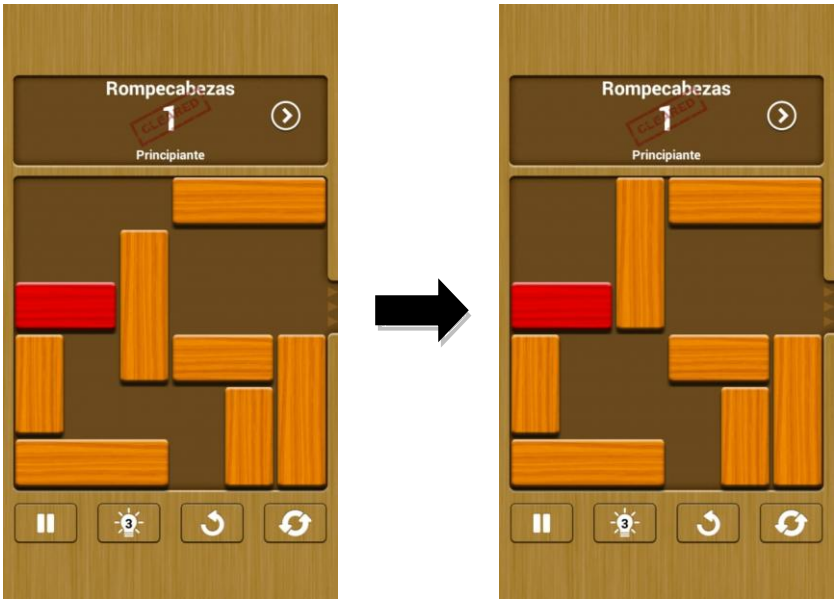

Imagen 2.9: ejemplo de solución de un panel. En la imagen de la izquierda, se ha desplazado una posición a la derecha la pieza de tamaño 3 horizontal de arriba. En la otra imagen, el jugador ha subido una posición la pieza vertical de tamaño 3 que está en la parte central del panel.

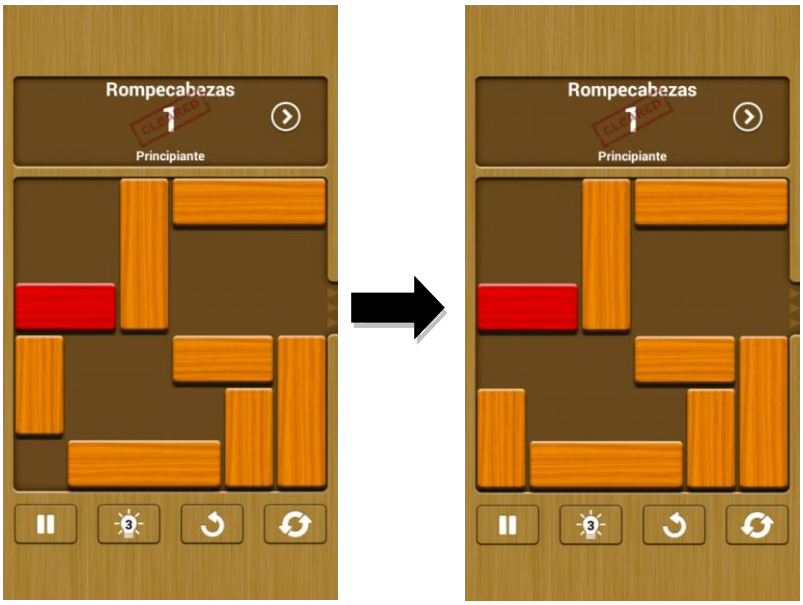

Imagen 2.10: ejemplo de solución de un panel. En la primera imagen, el jugador ha movido la pieza de tamaño 3 horizontal de la parte inferior del panel una posición a la derecha. En la segunda, se ha desplazado una posición hacia debajo la pieza de tamaño 2 vertical pegada a la parte izquierda del panel.

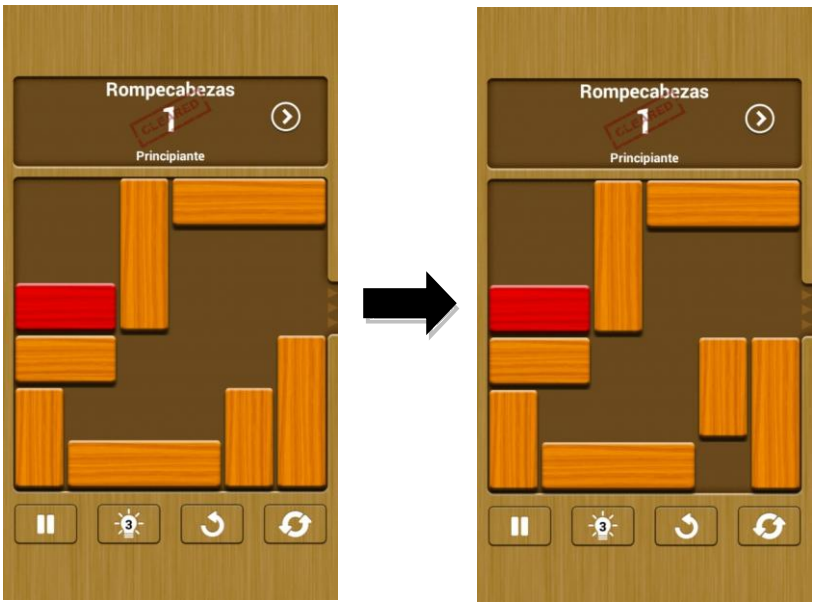

Imagen 2.11: ejemplo de solución de un panel. En este caso, el jugador ha desplazado, en la imagen de la izquierda, la pieza horizontal situada en el centro del panel de tamaño 2, a la posición más a la izquierda posible, en concreto justo debajo de la pieza objetivo de color rojo. En la imagen de la derecha, ha subido una posición la pieza vertical de tamaño 2, emparejándola con el principio de la pieza de tamaño 3 que tiene a su derecha.

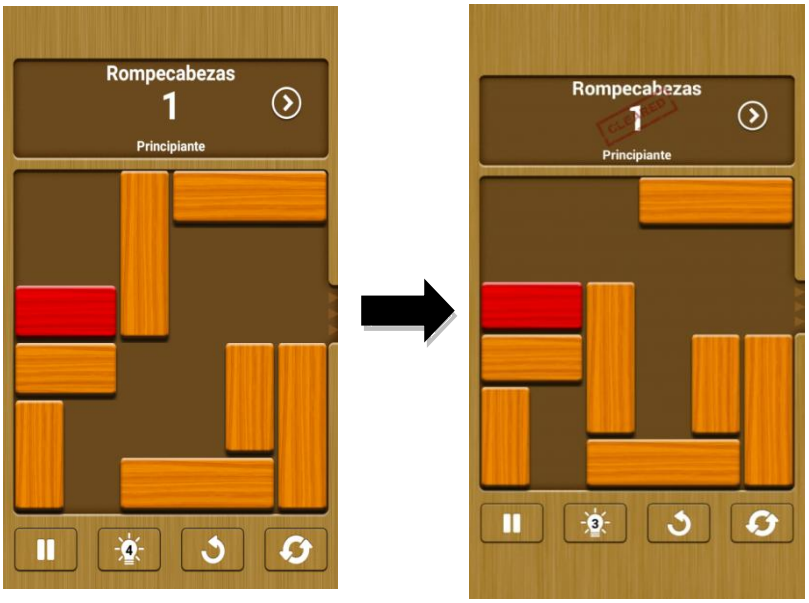

Imagen 2.12: ejemplo de solución de un panel. En la imagen de la izquierda se ha desplazado la pieza horizontal situada más abajo una posición a la derecha. En la imagen de la derecha, se ha bajado dos posiciones la pieza de tamaño 3 situada en el centro del panel.

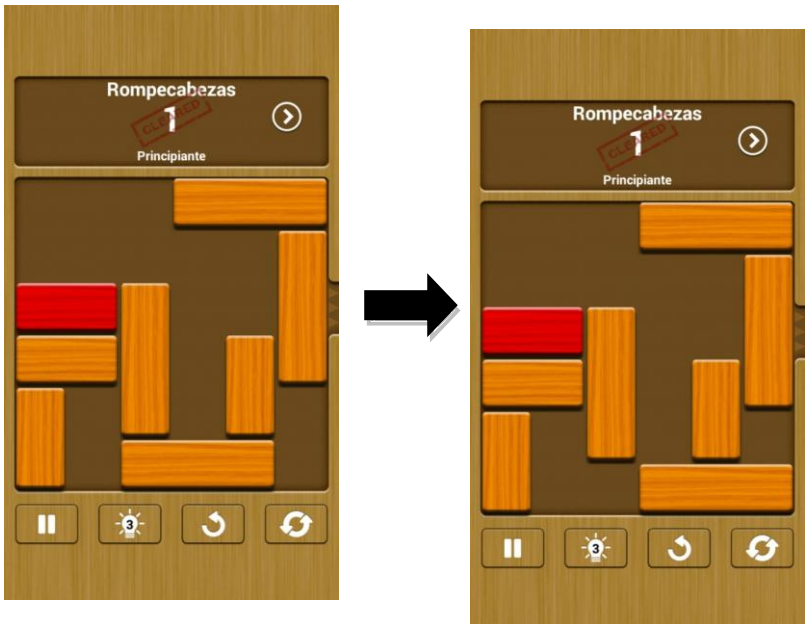

Imagen 2.13: ejemplo de solución de un panel. En este caso, en la primera imagen se ha desplazado la pieza de tamaño tres horizontal situada más a la derecha dos posiciones hacia arriba. En la otra imagen, se ha desplazado la pieza horizontal situada abajo del panel una posición a la derecha.

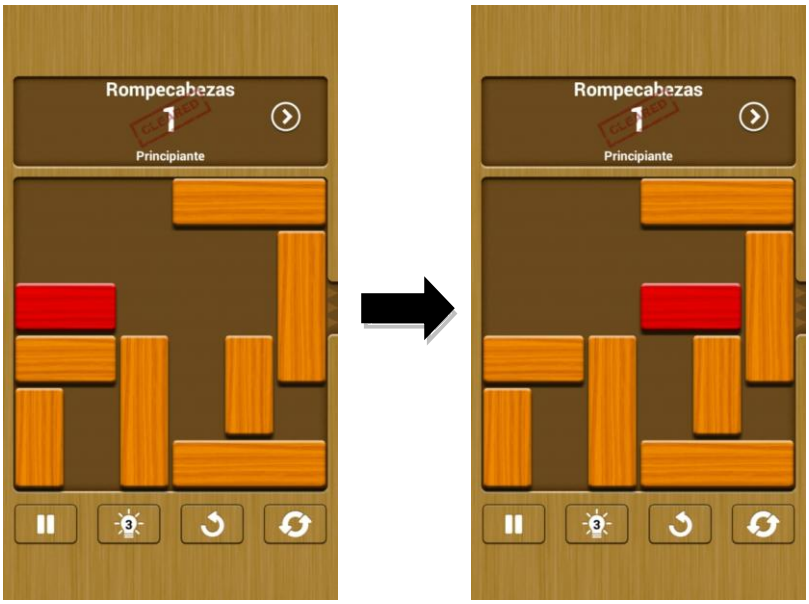

Imagen 2.14: ejemplo de solución de un panel. A continuación, se desplaza la pieza central vertical de tamaño 3 hacia abajo una posición. Después, en la imagen de la derecha, se desplaza la pieza objetivo de color rojo 3 posiciones a la derecha.

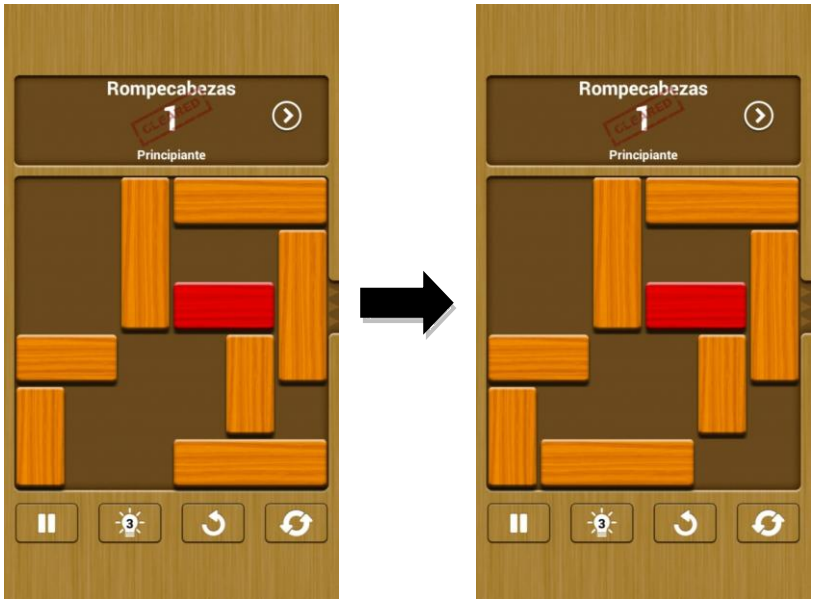

Imagen 2.15: ejemplo de solución de un panel. En la imagen de la izquierda, se desplaza la pieza central vertical hacia arriba. Sin embargo, en la imagen de la derecha, se desplaza la figura horizontal de debajo del panel hacia la izquierda dos posiciones.

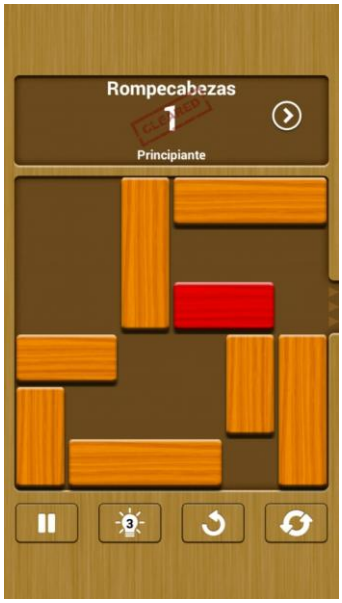

Imagen 2.16: ejemplo de solución de un panel. En la imagen podemos ver cómo se ha desplazado la pieza horizontal que estaba ocupando la puerta de salida hacia abajo, dejando libre el camino para que pueda salir la pieza de color rojo y completando así el panel.

3. FORMULACIÓN MATEMÁTICA DEL JUEGO

En este tercer capítulo se va a describir cada uno de los datos, variables, índices y restricciones que se emplearán en la formulación matemática del problema que se nos presenta.

Modelar en el ámbito de la programación matemática corresponde a expresar, mediante relaciones matemáticas, las especificaciones de un sistema u organización que gestiona unos recursos para realizar actividades siguiendo un determinado criterio de eficiencia.

El modelo genera una serie de expresiones y relaciones matemáticas que recibe el nombre de modelo de programación matemática.

Las especificaciones de un sistema son todas las características y normas determinadas que lo definen. Los recursos son los participantes que intervienen en el sistema. Los participantes son las entidades con las que trabaja el sistema y pueden ser tangibles o intangibles.

Las variables de decisión son las acciones controladas que se producen en el mismo, y que se definen sobre los participantes del sistema.

El criterio de eficiencia en el sistema es la dirección para perseguir un objetivo de mejora.

El modelo matemático está formado por tres componentes principales:

- Un conjunto de variables de decisión que se desarrollan en el sistema que se quiere optimizar. Pueden ser continuas, si toman valores reales e identifican medidas, o enteras, que son entidades discretas. Dentro de las variables enteras encontramos las binarias, las cuales únicamente toman los valores 1 y 0. Éstas sirven para la toma de decisiones que realice el modelo.

- Un conjunto de restricciones, que definen las especificaciones, características y normas del sistema.

- Una función objetivo, que es el criterio que se desea optimizar.

El formato general de un modelo lo podemos apreciar en la imagen 3.1:

$$\begin{aligned} &Min \quad f(x) \\ &sujeto\ a \\ &G_i(x) \approx b \quad i = 1...m \end{aligned}$$

Imagen 3.1: modelo general

Donde x representa un vector de variables, el signo ≈ hace referencia al signo de las restricciones que pueden ser: ≤; <; =; >; ≥. b es un vector de términos independientes, $G_i(x)\approx b$ es un conjunto de m restricciones y f(x) es la función objetivo.

A continuación, vamos a describir en mayor profundidad cada uno de los términos empleados para la formulación matemática del modelo.

3.1. Elementos del problema

Es importante para la implementación posterior en LINGO, la identificación correcta de todos los elementos que participan en el problema, así como las características de cada uno de dichos elementos.

Los elementos del problema pueden ser de cualquier naturaleza, además que también pueden representar recursos de espacio o tiempo.

Se distinguen por participar en las acciones que se producen en el sistema. Pueden aportar información en las especificaciones del mismo.

Una descripción explícita del sistema es necesaria para la identificación correcta de los participantes. En ocasiones, dicha descripción puede identificar otros participantes que posteriormente no tengan intervención en el problema de optimización que se plantee. A pesar de ello, su identificación no va a entorpecer la construcción del modelo.

Los participantes del problema poseen características simples que se utilizan en las especificaciones del sistema y en el criterio de optimización. El valor de una característica puede ser continuo, entero o binario.

A partir de lo descrito anteriormente, en el problema a resolver encontramos los siguientes participantes:

Tabla 3.1: resumen de los participantes en el problema.

Elementos	Conjunto e índice	Características
Piezas	Conjunto i=1…NP	- Tamaño (T_i) - Posición inicial (I_{ij}) - Fila o columna máxima donde se puede situar (C_i) - Posición (P_{ijt}) - Horizontal (H_i) - Fila inicial donde se encuentra cada pieza si la pieza es horizontal (F_i) - Columna inicial donde se encuentra cada pieza si la pieza es vertical (K_i)
Panel		- Dimensión (D)
Período	Conjunto t=0…NT	P_{ijt}
Filas	Conjunto j=1…6	P_{ijt} I_{ij}
Columnas	Conjunto j=1…6	P_{ijt} I_{ij}

En cuanto a los elementos del problema, para cada uno de los puzles empleados en este trabajo, destacamos los siguientes:

- Panel: Es un dato que nos permite conocer la dimensión de cada puzle. En el caso que nos ocupa, todos los paneles cuentan con la misma dimensión, en concreto de 6x6, por lo que no se utiliza como un dato en sí, pero sí se tiene en cuenta para el desarrollo de los otros datos, como pueden ser el número de filas o de columnas en las que se encuentran las piezas.
- Piezas: forman un conjunto de datos (y se representan por el índice i) que se caracterizan por un tamaño, denominado T_i, una posición inicial, I_{ij}, una posición para cada período, P_{ijt}, un dato que representa si la pieza es horizontal, valor 1, o vertical, valor 0, H_i, y otro dato que indica la fila o la columna donde se encuentra la pieza, dependiendo si es horizontal o vertical, F_i y K_i respectivamente.

- Período: no es un dato que se pueda extraer de los puzles, pero sí es un número que se ajusta manualmente en el problema. Se representa mediante el índice t.
- Filas: es un dato que se extrae a partir de la dimensión de los paneles. En este caso, tomará un valor máximo de 6 y se representa mediante el índice j.
- Columnas: al igual que las filas, es un dato fijado por la dimensión de los paneles. Se utiliza el mismo índice j que para las filas, ya que, si la pieza es horizontal, sólo cambiaría el valor de la columna en la que se encuentra la pieza, estando fija la fila en la que se encuentra. En caso de que la pieza fuese vertical, ocurrirá justamente lo contrario, es decir, que el único valor que variaría sería el de la fila donde se encuentra la pieza.

A continuación, vamos a describir cada uno de los valores que pueden tomar las características de los datos anteriormente descritos.

NP hace referencia al número total de piezas que tiene cada panel. En los puzzles que se han escogido para desarrollar este trabajo, NP varía desde un mínimo de 8 piezas hasta un máximo de 12.

Otro dato utilizado en la programación matemática del problema es el tamaño de cada pieza. Se define T_i a la dimensión que ocupa la pieza i. Este dato puede tomar los valores de 2 y 3.

En el caso de C_i, el cual hace referencia a la fila o columna máxima donde se puede encontrar una pieza, siempre va a tomar el valor máximo de 5, siendo ésta la posición mayor donde se puede colocar la primera parte de cada pieza, en el caso de que su tamaño sea de 2, y un valor máximo de 4, si su tamaño es 3.

H_i es un dato que se puede extraer de cada una de las piezas que conforman cada panel, y hace referencia a si la pieza es vertical u horizontal. Su valor será 1 si la pieza es horizontal y 0 si la pieza es vertical.

F_i es la fila donde se encuentra una pieza horizontal. Es decir, si la pieza es horizontal, siempre se va a encontrar en la misma fila pero puede cambiar de columna si se mueve.

Al igual que el anterior, K_i es la columna donde se encuentra una pieza vertical, ya que tomará un valor fijo para la columna, pudiendo cambiar la posición respecto de la fila.

A continuación, en la imagen 3.2 ejemplificamos lo descrito anteriormente para su mejor comprensión:

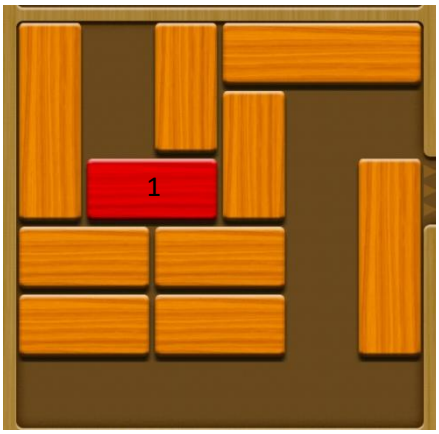

Imagen 3.2: ejemplo explicativo para los datos H_i, F_i y K_i

En este caso, para la pieza de color rojo, a la que se le asigna siempre i=1, H_1=1, ya que es una pieza horizontal, por lo que el valor de K_1 variará en función de la columna en la que se encuentre. Pero el valor de F_1 siempre será 3 ya que no puede cambiar la fila en la que se encuentra.

NT es el número máximo de iteraciones en las que se va a resolver el problema. Este dato se ha impuesto en el problema como dato, aunque se ha realizado un estudio, resolviendo cada problema para distintos NT y utilizando el menor NT con el que se consigue una solución óptima y alcanzable.

También se utiliza un número N, que toma un valor alto, y aparece en la función objetivo para asegurarse llegar al óptimo del problema. A la hora de resolver los modelos matemáticos en LINGO, se le ha dado a N el valor de 300.

Se define I_{ij} como la posición inicial que ocupa cada pieza. En realidad, hace referencia a la posición en la que se encuentra la primera parte de la pieza. Para entenderlo mejor, en la imagen 3.3 se ha señalado la posición de la pieza número 1 de color amarillo:

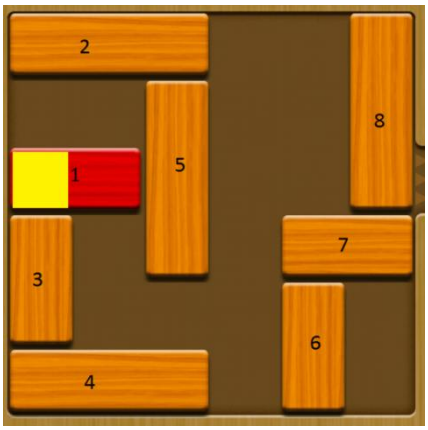

Imagen 3.3: ejemplo explicativo del dato I_{ij}

En el ejemplo anterior, el dato en concreto sería $I_{13} = 1$ y los demás I_{1j} toman el valor de 0, ya que una pieza solo puede estar en una única posición inicial.

3.2. Variables de decisión

La identificación de una variable está asociada a alguna acción en la que intervienen los elementos del problema.

Una vez identificada la variable, es necesario:

1. Identificar qué elementos del problema están involucrados en la acción.

   Se crea una variable por cada acción que se produce en el sistema con cada dato que intervenga en esa acción.

2. Definir qué tipo de valor toma esa variable.

   Existen 3 tipos de valores:

   - Continuo. La variable de decisión se define como una variable continua.
   - Entero. La variable de decisión se define como una variable entera.
   - Binario. La variable de decisión se corresponde con una decisión de hacer o no una acción, o los valores se definen como cardinales sobre una serie de participantes.

En el caso que nos concierne, encontramos la variable de decisión de mover, que incluye a los participantes piezas, filas o columnas, y período. Su valor sería binario, siendo 1 si la pieza i en el período t se encuentra en la fila o columna j, y 0 si no. Por tanto, se genera la variable α_{ijt}.

En el segundo modelo empleado en este trabajo, también se genera otra variable, en concreto la llamamos $MOVE_{ijt}$, que también es de tipo binario y tomará el valor 1 si se realiza el movimiento de la pieza i de la posición j a j' desde el período t a t+1; y tomará el valor 0 si no.

3.3. Especificaciones

En este apartado se van a describir cada una de las restricciones que forman parte del modelo matemático.

Para el primer modelo, se tienen en cuenta un conjunto de 4 restricciones, las cuales se desarrollarán a continuación.

El primer conjunto de restricciones hace referencia a que cada pieza se encuentra en una única posición inicial. Por tanto, este conjunto de restricciones tendrá tantas restricciones como piezas tenga el problema.

Así, dichas restricciones vienen definidas tal que:

$$\alpha_{i\_j\_0} = 1 \quad \forall i;\ \forall j\ /\ I_{ij} = 1;$$

Para entender mejor este conjunto de restricciones, se va a ejemplificar con las correspondientes al panel 3 del nivel principiante, el cual se puede apreciar en la imagen 3.4:

Imagen 3.4: puzle 3 del nivel principiante

Así, las especificaciones serían:

$\alpha_{1\_1\_0} = 1;$

$\alpha_{2\_1\_0} = 1;$

$\alpha_{3\_1\_0} = 1;$

$\alpha_{4\_2\_0} = 1;$

$\alpha_{5\_2\_0} = 1;$

$\alpha_{6\_4\_0} = 1;$

$\alpha_{7\_3\_0} = 1;$

$\alpha_{8\_3\_0} = 1;$

El siguiente grupo de restricciones nos asegura que en cada período t, la pieza i esté en una posición j determinada, de tal manera que:

$\sum_{j=1}^{C_i} \alpha_{i\_j\_t} = 1 \quad \forall i;\ \forall t\ ;$

En el caso del panel 3, fijado un período $t\ = 1$, las restricciones serían:

$\alpha_{1\_1\_1} + \alpha_{1\_2\_1} + \alpha_{1\_3\_1} + \alpha_{1\_4\_1} + \alpha_{1\_5\_1} = 1;$

$\alpha_{2\_1\_1} + \alpha_{2\_2\_1} + \alpha_{2\_3\_1} + \alpha_{2\_4\_1} + \alpha_{2\_5\_1} = 1;$

$\alpha_{3\_1\_1} + \alpha_{3\_2\_1} + \alpha_{3\_3\_1} + \alpha_{3\_4\_1} = 1;$

$\alpha_{4\_1\_1} + \alpha_{4\_2\_1} + \alpha_{4\_3\_1} + \alpha_{4\_4\_1} + \alpha_{4\_5\_1} = 1;$

$\alpha_{5\_1\_1} + \alpha_{5\_2\_1} + \alpha_{5\_3\_1} + \alpha_{5\_4\_1} + \alpha_{5\_5\_1} = 1;$

$\alpha_{6\_1\_1} + \alpha_{6\_2\_1} + \alpha_{6\_3\_1} + \alpha_{6\_4\_1} + \alpha_{6\_5\_1} = 1;$

$\alpha_{7\_1\_1} + \alpha_{7\_2\_1} + \alpha_{7\_3\_1} + \alpha_{7\_4\_1} = 1;$

$\alpha_{8\_1\_1} + \alpha_{8\_2\_1} + \alpha_{8\_3\_1} + \alpha_{8\_4\_1} = 1;$

A continuación, se describe un conjunto de restricciones para que una pieza i no pueda saltar a otras piezas y, además, solo pueda realizar como máximo un movimiento desde un período t-1 a otro período t. Este grupo de restricciones quedaría de la siguiente manera:

$\alpha_{i\_j\_t-1} + \alpha_{i\_j'\_t} \leq 1\ \forall i;\ \forall t;\ \forall j\ /\ j = 1 \dots (7 - T_i);$

$\forall j'/\ j' = 1 \dots (7 - T_i);\quad j \neq j';\ j' > j + 1;$

Para el puzle 3, algunas de las restricciones serían:

$\alpha_{1\_1\_0} + \alpha_{1\_3\_1} \leq 1;$

$\alpha_{2\_2\_0} + \alpha_{2\_4\_1} \leq 1;$

$\alpha_{3\_4\_0} + \alpha_{3\_1\_1} \leq 1;$

$\alpha_{4\_5\_0} + \alpha_{4\_2\_1} \leq 1;$

$\alpha_{5\_3\_0} + \alpha_{5\_5\_1} \leq 1;$

$\alpha_{6\_4\_0} + \alpha_{6\_1\_1} \leq 1;$

$\alpha_{7\_2\_0} + \alpha_{7\_4\_1} \leq 1;$

$\alpha_{8\_1\_0} + \alpha_{8\_3\_1} \leq 1;$

El último grupo de restricciones que presenta el primer modelo, consigue que dos piezas distintas, i e i', no puedan cruzarse ni superponerse, es decir, que si una pieza

quiere realizar un movimiento a una posición que ya está ocupada, no lo podrá realizar. Así, dicha restricción se vería reflejada en el modelo tal que así:

$\alpha_{i\_j\_t} + \alpha_{i\prime\_j\prime\_t} \leq 1 \ \forall t; \ \forall i; \ \forall i' / P_{i\_j\_t} \neq P_{i\prime\_j\prime\_t} \ y \ (H_i = 1 \ y \ H_{i\prime} = 0)$ ó

$(H_i = 0 \ y \ H_{i\prime} = 1); \ \forall j; \ \forall j' / (j + T_i \geq K_{i'} \ y \ j \leq K_{i'})$ ó $(j' + T_{i'} \geq F_i \ y \ j' \leq F_i);$

En el caso que se ha escogido para ejemplificar las restricciones, algunas de ellas quedarían tal que así:

$\alpha_{1\_3\_1} + \alpha_{7\_2\_1} \leq 1;$

$\alpha_{2\_4\_1} + \alpha_{3\_1\_1} \leq 1;$

$\alpha_{3\_4\_1} + \alpha_{7\_4\_1} \leq 1;$

$\alpha_{4\_2\_1} + \alpha_{6\_3\_1} \leq 1;$

$\alpha_{5\_5\_1} + \alpha_{3\_3\_1} \leq 1;$

$\alpha_{6\_1\_1} + \alpha_{2\_1\_1} \leq 1;$

$\alpha_{7\_2\_1} + \alpha_{1\_3\_1} \leq 1;$

$\alpha_{8\_3\_1} + \alpha_{1\_5\_1} \leq 1;$

En cuanto al segundo modelo, el número de restricciones aumenta hasta 5, con una única diferencia, que es que se le añade un nuevo conjunto de restricciones que se va a explicar seguidamente.

El conjunto de restricciones nuevo que habría que añadir al modelo anterior asegura que si la pieza i cambia de posición j de un período t-1 a otro período t, la variable MOVE tome el valor de 1. Por tanto, se vería reflejado de la siguiente forma:

$\alpha_{i\_j\_t-1} - \alpha_{i\_j\_t} \leq MOVE_{i\_j\_t} \ \forall i; \ \forall j; \ \forall t;$

En el puzle 3, algunas de estas restricciones quedarían tal que:

$\alpha_{1\_3\_0} - \alpha_{1\_3\_1} \leq MOVE_{1\_3\_1};$

$\alpha_{2\_1\_0} - \alpha_{2\_1\_1} \leq MOVE_{2\_1\_1};$

$\alpha_{3\_2\_0} - \alpha_{3\_2\_1} \leq MOVE_{3\_2\_1};$

$\alpha_{4\_4\_0} - \alpha_{4\_4\_1} \leq MOVE_{4\_4\_1};$

$\alpha_{5\_2\_0} - \alpha_{5\_2\_1} \leq MOVE_{5\_2\_1};$

$\alpha_{6\_5\_0} - \alpha_{6\_5\_1} \leq MOVE_{6\_5\_1};$

$\alpha_{7\_3\_0} - \alpha_{7\_3\_1} \leq MOVE_{7\_3\_1}$;

$\alpha_{8\_1\_0} - \alpha_{8\_1\_1} \leq MOVE_{8\_1\_1}$;

3.4. Función objetivo

En la función objetivo se expresan todos los costes de las variables de decisión que intervienen en el problema, ya sean positivos o negativos.

Por tanto, la función objetivo ayuda a identificar las variables de decisión del sistema, pues toda acción que conlleve un coste se corresponderá con una variable.

La situación normal es que el coste unitario de una variable no varíe sea cual sea el valor de la actividad.

Si la variable es binaria, conllevará un coste c si toma el valor 1, y no conllevará coste si su valor es 0.

Sin embargo, para variables enteras o continuas, el coste que se aplica puede depender del valor que toma la variable.

En cuanto al criterio de eficiencia, podemos escoger entre maximizar la función objetivo o minimizarla, según queramos obtener el mayor o menor valor de la función a partir de un dominio de valores que puede tomar dicha función.

El primer modelo lo que pretende hacer es que, como máximo, en el último movimiento, la pieza objetivo de color rojo tenga el camino despejado para poder salir del panel. Para ello, se utiliza una función objetivo de maximizar un número N, que para poder resolver el modelo en LINGO se le ha dado un valor de 300, por la variable de decisión $\alpha_{1\_5\_NT}$, tal y como podemos observar a continuación:

$MAX\ N \times \alpha_{1\_5\_NT}$

En cuanto al segundo modelo empleado, la única diferencia que se observa sería la inclusión de una nueva variable en la función objetivo, que contabiliza los movimientos de todas las piezas que se han realizado durante la resolución del modelo, así como también aparecería en una restricción, para que se tengan en cuenta cada uno de dichos movimientos.

Así, la nueva función objetivo se expresaría de la siguiente forma:

$$MAX\left[N \times \alpha_{1\_5\_NT} - \left(\sum_{t=1}^{NT}\sum_{i=1}^{NP}\sum_{j=1}^{7-T_i} MOVE_{i\_j\_t}\right)\right]$$

3.5. Modelos matemáticos

Seguidamente, se va a describir cada uno de los modelos, de forma abstracta, que se han utilizado para desarrollar este trabajo fin de grado.

Por tanto, escribiendo el primer modelo de forma abstracta, se obtendría lo siguiente:

$MAX\ N \times \alpha_{1\_5\_NT}$

s.a:

1) $\alpha_{i\_j\_0} = 1 \quad \forall i;\ \forall j\ /\ I_{ij} = 1;$

2) $\sum_{j=1}^{7-T_i} \alpha_{i\_j\_t} = 1 \quad \forall i;\ \forall t\ ;$

3) $\alpha_{i\_j\_t-1} + \alpha_{i\_j'\_t} \leq 1\ \forall i;\ \forall t;\ \forall j\ /\ j = 1 \dots (7 - T_i);$

$\forall j'/\ j' = 1 \dots (7 - T_i);\quad j \neq j';\ j' > j + 1;$

4) $\alpha_{i\_j\_t} + \alpha_{i'\_j'\_t} \leq 1\ \forall t;\ \forall i;\ \forall i'/\ P_{i\_j\_t} \neq P_{i'\_j'\_t}\ y\ (H_i = 1\ y\ H_{i'} = 0)$ ó

$(H_i = 0\ y\ H_{i'} = 1);\ \forall j;\ \forall j'/(\ j + T_i \geq K_{i'}\ y\ j \leq K_{i'})$ ó $(j' + T_{i'} \geq F_i\ y\ j' \leq F_i);$

$\alpha_{i\_j\_t}\ binaria;\ i = 1 \dots NP;\ i' = 1 \dots NP;\ j = 1 \dots (7 - T_i);\ j' = 1 \dots (7 - T_i);$

$t = 1 \dots NT;$

Asimismo, el segundo modelo empleado en la resolución de los distintos paneles está descrito de la siguiente forma:

$$MAX\left[N \times \alpha_{1\_5\_NT} - \left(\sum_{t=1}^{NT}\sum_{i=1}^{NP}\sum_{j=1}^{7-T_i} MOVE_{i\_j\_t}\right)\right]$$

s.a:

1) $\alpha_{i\_j\_t-1} - \alpha_{i\_j\_t} \leq MOVE_{i\_j\_t}\ \forall i;\ \forall j;\ \forall t;$

2) $\alpha_{i\_j\_0} = 1\ \ \forall i;\ \forall j\ /\ I_{ij} = 1;$

3) $\sum_{j=1}^{7-T_i} \alpha_{i\_j\_t} = 1\ \ \ \forall i;\ \forall t\ ;$

4) $\alpha_{i\_j\_t-1} + \alpha_{i\_j'\_t} \leq 1\ \forall i;\ \ \forall t;\ \ \forall j\ /\ j = 1 \ldots (7 - T_i);$

$\forall j'\ /\ j' = 1 \ldots (7 - T_i);\ \ j \neq j';\ j' > j + 1;$

5) $\alpha_{i\_j\_t} + \alpha_{i'\_j'\_t} \leq 1\ \forall t;\ \forall i;\ \forall i'\ /\ P_{i\_j\_t} \neq P_{i'\_j'\_t}\ y\ (H_i = 1\ y\ H_{i'} = 0)$ ó

$(H_i = 0\ y\ H_{i'} = 1);\ \forall j;\ \forall j'/(\ j + T_i \geq K_{i'}\ y\ j \leq K_{i'})$ ó $(j' + T_{i'} \geq F_i\ y\ j' \leq F_i);$

$\alpha_{i\_j\_t}\ binaria;\ i = 1 \ldots NP;\ i' = 1 \ldots NP;\ j = 1 \ldots (7 - T_i);\ j' = 1 \ldots (7 - T_i);$

$t = 1 \ldots NT;$

4. IMPLEMENTACIÓN DE LOS MODELOS

En este cuarto capítulo vamos a describir la implementación matemática de los dos modelos empleados para resolver el juego *Unblock Me Free*. Para ello, vamos a explicar, en primer lugar, las herramientas utilizadas, así como el lenguaje de programación utilizado.

4.1. Lenguaje de programación: Visual Basic

Para desarrollar los modelos matemáticos descritos en el apartado anterior en la herramienta LINGO, se ha utilizado un lenguaje de programación concreto, Visual Basic.

Es un lenguaje de programación dirigido por eventos, que fue desarrollado por Alan Cooper, cuya primera versión fue presentada en 1991, con la intención de simplificar la programación utilizando un ambiente de desarrollo.

Las características básicas de este programa se presentan de la siguiente forma:

- En la parte superior aparece la barra de título, donde se encuentra el nombre del proyecto, la barra de menú, que consta de 13 opciones, y la barra de herramientas.
- En la parte central se encuentra el espacio de trabajo. Aquí es donde se desarrolla el lenguaje de programación.

La estructura que sigue un proyecto en Visual Basic es la siguiente:

- Declaración de variables: se declaran las variables necesarias para realizar las funciones.
- Condiciones iniciales: son condiciones que afectarán a todo el proyecto.
- Funciones: la primera función que nos encontramos es la función principal, que se encarga de llamar a las demás funciones, para que se lean los datos del problema, así como para que se realicen todas las operaciones y cálculos necesarios.

En cuanto a las funciones, para realizar este proyecto se han utilizado las siguientes:

- Main: esta función es la principal, ya que se encarga de llamar a todas las demás funciones para que realicen los cálculos necesarios. En concreto, esta función trabaja de la siguiente forma:
    - Se declaran todas las variables que se vayan a utilizar para llevar a cabo las operaciones de la función.

- Se define el panel que se va a elegir para resolver.
- La función ObtenerDatos se encarga de conseguir los datos necesarios para resolver el modelo.
- Llama a la función ModelarPosiciones.
- Llama a la función ModelarPosicionesOptimo.

El código de la función main es el siguiente:

```
Sub main()
Dim I As Integer
Dim J As Integer
Dim K As Integer
Dim Num As Integer
Panel = "P1"
ObtenerDatos
NT = 10
ModelarPosiciones
ModelarPosicionesOptimo
End Sub
```

- ObtenerDatos: su función es la de obtener los datos del problema a partir de archivos de textos. Su estructura es la siguiente:
  - Se declaran todas las variables necesarias para poder llevar a cabo las operaciones de esta función.
  - Se abre el archivo de texto del cual se van a extraer cada uno de los datos necesarios para generar el modelo.
  - Se asignan los datos a variables globales declaradas.

Así, el código de la función ObtenerDatos en Visual Basic es:

```
Sub ObtenerDatos()

Dim I As Integer

Dim J As Integer

Dim Canal As Integer

Dim Linea As String

Canal = FreeFile

Open App.Path & "\TABLEROS\principiante\" & Panel & ".txt" For Input
As Canal

Line Input #Canal, Linea

NumFichas = Linea

ReDim Fichas(NumFichas)

Line Input #Canal, Linea

NumFilas = Linea

Line Input #Canal, Linea

NumColumnas = Linea

ReDim Tablero(NumFilas, NumColumnas)

Line Input #Canal, Linea

For I = 1 To NumFichas

Line Input #Canal, Linea

Fichas(I).Posicion = Linea

Next

Line Input #Canal, Linea

For I = 1 To NumFichas

Line Input #Canal, Linea

Fichas(I).Tamano = Linea

Next
```

```
Line Input #Canal, Linea
For I = 1 To NumFichas
Line Input #Canal, Linea
Fichas(I).Ref = Linea
Next
Line Input #Canal, Linea
For I = 1 To NumFilas
Line Input #Canal, Linea
For J = 1 To NumColumnas
If J < NumColumnas Then
Tablero(I, J) = Left(Linea, InStr(Linea, " "))
Linea = Right(Linea, Len(Linea) - InStr(Linea, " "))
Else
Tablero(I, J) = Linea
End If
Next
Next
End Sub
```

- ModelarPosiciones: su función es crear el primer modelo empleado para la resolución de los paneles que se quieren resolver posteriormente en LINGO. Esta función sigue la siguiente estructura:
    - En primer lugar, se definen las variables que se van a emplear en el modelo.
    - Se abre el archivo de extensión .lg4 donde se va a escribir el modelo matemático.
    - Se escribe la función objetivo.

- Se declaran las variables binarias del modelo.
- Se describe la restricción por la que cada pieza se encuentra en un lugar del panel al inicio.
- Se escribe la restricción mediante la cual se asegura que cada pieza del panel sólo puede estar en un sitio.
- Se escribe la restricción por la cual una pieza sólo puede realizar, como máximo, un movimiento de un período t-1 a otro t. Así, nos aseguramos que una pieza no pueda saltar otras piezas.
- Se escribe la restricción que nos asegura que si el sitio está ocupado por otra pieza, no se puede realizar movimiento, en ese período, a ese lugar. Para ello, se asegura que dos piezas no puedan cruzarse ni superponerse.

El código de esta función desarrollado en Visual Basic quedaría tal que así:

```
Public Sub ModelarPosiciones()
Dim Canal As Integer
Dim Cadena As String
Dim I As Integer
Dim J As Integer
Dim P As Integer
Dim PP As Integer
Dim K As Integer
Dim KK As Integer
Dim T As Integer
Dim Cadena1 As String
Dim Cadena2 As String
Dim Cadena3 As String
Dim Cadena4 As String
Canal = FreeFile
Open App.Path & "\" & Panel & ".lg4" For Output As Canal
```

```
Cadena = "MAX = " & "300*ALFA1_5" & "_" & NT & ";"
Print #Canal, Cadena
Print #Canal, ""
For T = 1 To NT
 For K = 1 To NumFichas
Cadena = ""
For P = 2 To (7 - Fichas(K).Tamano) - 1
Cadena = "@Bin(" & "ALFA" & K & "_" & P & "_" & T & ");"
Print #Canal, Cadena
Next
Next
Next
For K = 1 To NumFichas
If Fichas(K).Posicion = 1 Then
For J = 1 To NumColumnas
If Tablero(Fichas(K).Ref, J) = K Then
Cadena = "ALFA" & K & "_" & J & "_" & 0 & "=1;"
Print #Canal, Cadena
Exit For
End If
Next
Else
For I = 1 To NumFilas
If Tablero(I, Fichas(K).Ref) = K Then
Cadena = "ALFA" & K & "_" & I & "_" & 0 & "=1;"
Print #Canal, Cadena
Exit For
```

```
End If
Next
End If
Next
Print #Canal, ""
For T = 1 To NT
For K = 1 To NumFichas
Cadena = ""
For P = 1 To (7 - Fichas(K).Tamano)
Cadena = Cadena & "ALFA" & K & "_" & P & "_" & T & " + "
Next
Cadena = Left(Cadena, Len(Cadena) - 3)
Cadena = Cadena & "=1 ;"
Print #Canal, Cadena
Next
Next
Print #Canal, ""
Print #Canal, ""
For T = 1 To NT
For K = 1 To NumFichas
For P = 1 To (7 - Fichas(K).Tamano)
For PP = 1 To (7 - Fichas(K).Tamano)
If P < 5 And PP > P + 1 Then
Cadena = "ALFA" & K & "_" & P & "_" & T - 1 & " + " & "ALFA" & K &
"_" & PP          & "_" & T & "<=1;"
 Print #Canal, Cadena
 End If
```

```
If PP < 5 And P > PP + 1 Then

 Cadena = "ALFA" & K & "_" & P & "_" & T - 1 & " + " & "ALFA" & K & "_" & PP & "_" & T & "<=1;"

Print #Canal, Cadena

End If

Next

Next

Next

Next

Print #Canal, ""

For T = 1 To NT

For K = 1 To NumFichas

For KK = 1 To NumFichas

If Fichas(K).Posicion <> Fichas(KK).Posicion Then

For P = 1 To (7 - Fichas(K).Tamano)

For PP = 1 To (7 - Fichas(KK).Tamano)

If Fichas(K).Tamano + P - 1 >= Fichas(KK).Ref And P <= Fichas(KK).Ref And  Fichas(KK).Tamano + PP - 1 >= Fichas(K).Ref And PP <= Fichas(K).Ref Then

Cadena = "ALFA" & K & "_" & P & "_" & T & " + " & "ALFA" & KK & "_" & PP & "_" & T & " <=1;"

 Print #Canal, Cadena

 End If

 Next

 Next

 Else

 If Fichas(K).Ref = Fichas(KK).Ref And K <> KK Then

 For P = 1 To (7 - Fichas(K).Tamano)

 For PP = 1 To (7 - Fichas(KK).Tamano)
```

```
If P + Fichas(K).Tamano - 1 >= PP And P + Fichas(K).Tamano - 1 <=
PP + Fichas(KK).Tamano - 1 Then

Cadena = "ALFA" & K & "_" & P & "_" & T & " + " & "ALFA" & KK &
"_" & PP & "_" & T & " <=1;"

 Print #Canal, Cadena

 End If

 If PP + Fichas(KK).Tamano - 1 >= P And PP + Fichas(KK).Tamano -
1 <= P + Fichas(K).Tamano - 1 Then

 Cadena = "ALFA" & K & "_" & P & "_" & T & " + " & "ALFA" & KK &
"_" & PP & "_" & T & " <=1;"

 Print #Canal, Cadena

 End If

 Next

 Next

 End If

 End If

 Next

 Next

 Next

 Print #Canal, ""

Close Canal

End Sub
```

- ModelarPosicionesOptimo: su función es crear el segundo modelo empleado para la resolución de los paneles que se quieren resolver posteriormente en LINGO. Esta función sigue la siguiente estructura:
    - En primer lugar, se definen las variables que se van a emplear en el modelo.
    - Se abre el archivo de extensión .lg4 donde se va a escribir el modelo matemático.
    - Se escribe la función objetivo.

- Se declaran las variables binarias del modelo.
- Se escribe la restricción por la que, para una misma pieza, desde el período t-1 al período t, se activa la variable $MOVE_{ijt}$ si la pieza i ha cambiado de la posición j a j' en el período t.
- Se describe la restricción por la que cada pieza se encuentra en un lugar del panel al inicio.
- Se escribe la restricción mediante la cual se asegura que cada pieza del panel sólo puede estar en un sitio.
- Se escribe la restricción por la cual una pieza sólo puede realizar, como máximo, un movimiento de un período t-1 a otro t. Así, nos aseguramos que una pieza no pueda saltar otras piezas.
- Se escribe la restricción que nos asegura que si el sitio está ocupado por otra pieza, no se puede realizar movimiento, en ese período, a ese lugar. Para ello, se asegura que dos piezas no puedan cruzarse ni superponerse.

La función ModelarPosicionesOptimo en el lenguaje de Visual Basic quedaría de la siguiente forma:

```
Public Sub ModelarPosicionesOptimo()
Dim Canal As Integer
Dim Cadena As String
Dim I As Integer
Dim J As Integer
Dim P As Integer
Dim PP As Integer
Dim K As Integer
Dim KK As Integer
Dim T As Integer
Dim Cadena1 As String
Dim Cadena2 As String
Dim Cadena3 As String
```

```
Dim Cadena4 As String
   Canal = FreeFile
   Open App.Path & "\" & Panel & "_Opt" & ".lg4" For Output As Canal
   Cadena = "MAX = " & "300*ALFA1_5" & "_" & NT & " - ( "
   For T = 1 To NT
    For K = 1 To NumFichas
     For P = 1 To (7 - Fichas(K).Tamano)
        Cadena = Cadena & "MOVE" & K & "_" & P & "_" & T & " + "
     Next
    Next
   Next
   Cadena = Left(Cadena, Len(Cadena) - 3)
   Cadena = Cadena & ");"
   Print #Canal, Cadena
   Print #Canal, ""
   For T = 1 To NT
    For K = 1 To NumFichas
     Cadena = ""
     For P = 2 To (7 - Fichas(K).Tamano) - 1
        Cadena = "@Bin(" & "ALFA" & K & "_" & P & "_" & T & ");"
        Print #Canal, Cadena
     Next
    Next
   Next
   For T = 1 To NT
    For K = 1 To NumFichas
      For P = 1 To (7 - Fichas(K).Tamano)
```

```
        Cadena = "ALFA" & K & "_" & P & "_" & T - 1 & " - " & "ALFA" & K & "_" & P & "_" & T & " <= " & "MOVE" & K & "_" & P & "_" & T & ";"
        Print #Canal, Cadena
      Next
     Next
    Next
    Print #Canal, ""
    For K = 1 To NumFichas
     If Fichas(K).Posicion = 1 Then
      For J = 1 To NumColumnas
       If Tablero(Fichas(K).Ref, J) = K Then
        Cadena = "ALFA" & K & "_" & J & "_" & 0 & "=1;"
        Print #Canal, Cadena
        Exit For
       End If
      Next
     Else
      For I = 1 To NumFilas
       If Tablero(I, Fichas(K).Ref) = K Then
        Cadena = "ALFA" & K & "_" & I & "_" & 0 & "=1;"
        Print #Canal, Cadena
        Exit For
       End If
      Next
     End If
    Next
    Print #Canal, ""
```

```
    For T = 1 To NT
     For K = 1 To NumFichas
      Cadena = ""
      For P = 1 To (7 - Fichas(K).Tamano)
        Cadena = Cadena & "ALFA" & K & "_" & P & "_" & T & " + "
      Next
      Cadena = Left(Cadena, Len(Cadena) - 3)
      Cadena = Cadena & "=1 ;"
      Print #Canal, Cadena
     Next
    Next
    Print #Canal, ""
    Print #Canal, ""
    For T = 1 To NT
     For K = 1 To NumFichas
      For P = 1 To (7 - Fichas(K).Tamano)
       For PP = 1 To (7 - Fichas(K).Tamano)
        If P < 5 And PP > P + 1 Then
         Cadena = "ALFA" & K & "_" & P & "_" & T - 1 & " + " & "ALFA" & K
& "_" & PP & "_" & T & "<=1;"
         Print #Canal, Cadena
        End If
        If PP < 5 And P > PP + 1 Then
         Cadena = "ALFA" & K & "_" & P & "_" & T - 1 & " + " & "ALFA" & K
& "_" & PP & "_" & T & "<=1;"
         Print #Canal, Cadena
        End If
      Next
```

```
      Next
     Next
    Next
    Print #Canal, ""
    For T = 1 To NT
     For K = 1 To NumFichas
       For KK = 1 To NumFichas
        If Fichas(K).Posicion <> Fichas(KK).Posicion Then
         For P = 1 To (7 - Fichas(K).Tamano)
          For PP = 1 To (7 - Fichas(KK).Tamano)
           If Fichas(K).Tamano + P - 1 >= Fichas(KK).Ref And P <= Fichas(KK).Ref And Fichas(KK).Tamano + PP - 1 >= Fichas(K).Ref And PP <= Fichas(K).Ref Then
             Cadena = "ALFA" & K & "_" & P & "_" & T & " + " & "ALFA" & KK & "_" & PP & "_" & T & " <=1;"
             Print #Canal, Cadena
           End If
          Next
         Next
        Else
         If Fichas(K).Ref = Fichas(KK).Ref And K <> KK Then
          For P = 1 To (7 - Fichas(K).Tamano)
           For PP = 1 To (7 - Fichas(KK).Tamano)
            If P + Fichas(K).Tamano - 1 >= PP And P + Fichas(K).Tamano - 1 <= PP + Fichas(KK).Tamano - 1 Then
             Cadena = "ALFA" & K & "_" & P & "_" & T & " + " & "ALFA" & KK & "_" & PP & "_" & T & " <=1;"
             Print #Canal, Cadena
            End If
```

```
            If PP + Fichas(KK).Tamano - 1 >= P And PP + Fichas(KK).Tamano - 1 <= P + Fichas(K).Tamano - 1 Then

                Cadena = "ALFA" & K & "_" & P & "_" & T & " + " & "ALFA" & KK & "_" & PP & "_" & T & " <=1;"

                Print #Canal, Cadena

            End If

          Next

         Next

        End If

       End If

      Next

    Next

   Next

   Print #Canal, ""

  Close Canal

End Sub
```

A continuación, vamos a mostrar las diferentes pantallas que se pueden apreciar en Visual Basic.

Al abrir el programa, aparece una pantalla principal, en la que se incluye el menú principal, como se muestra en la imagen 4.1:

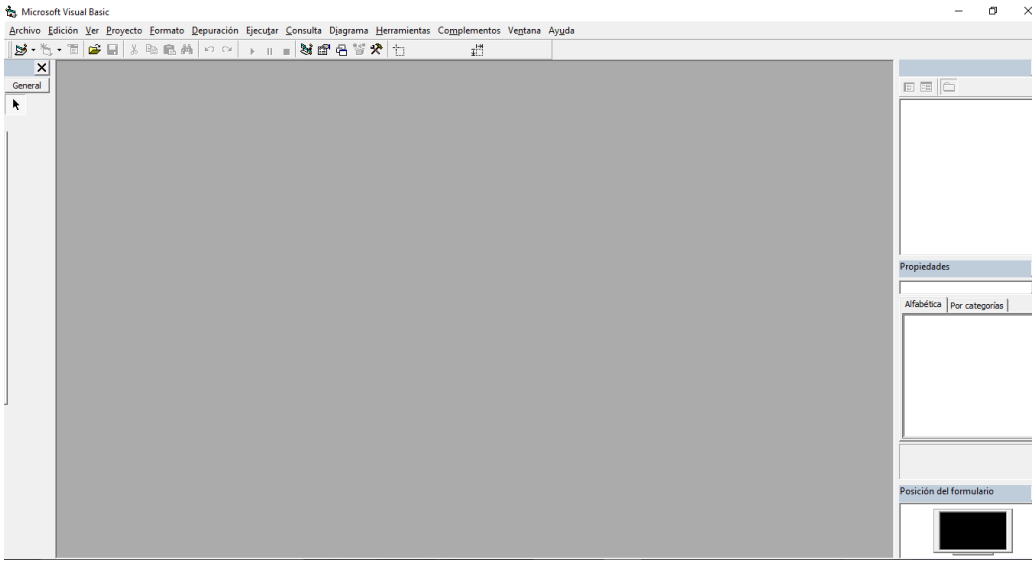

Imagen 4.1: pantalla principal Visual Basic

Una vez que se ha abierto el programa, para abrir un proyecto que ya se ha creado, en el menú principal, en la pestaña Archivo, le damos a Abrir proyecto y buscamos el proyecto que queremos, tal y como se muestra en la imagen 4.2:

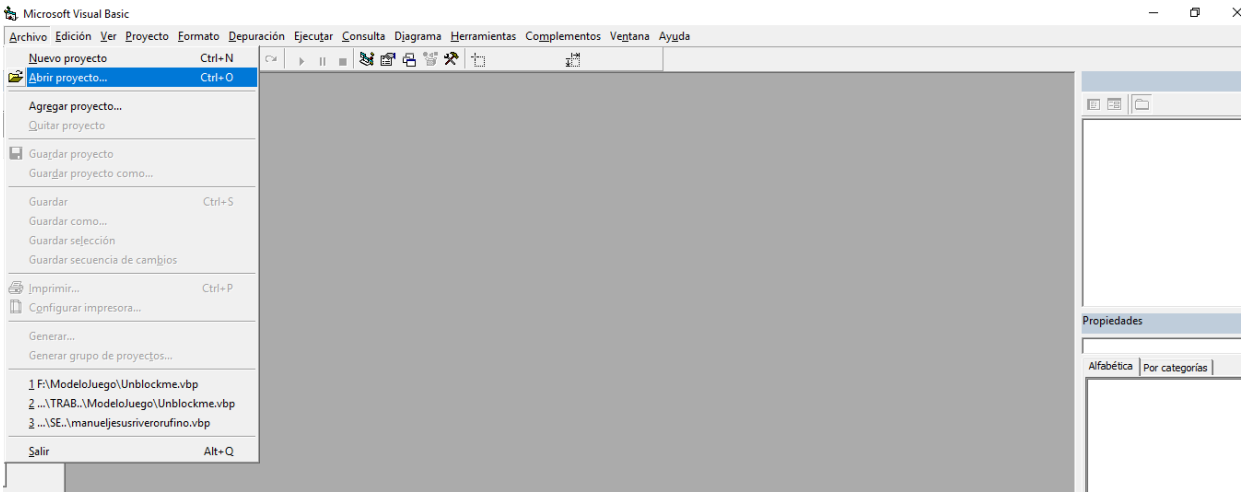

Imagen 4.2: abrir proyecto

Cuando hayamos seleccionado el proyecto que queramos abrir, se desplegará una ventana en la que aparece el código del proyecto. En la imagen 4.3 podemos ver un ejemplo:

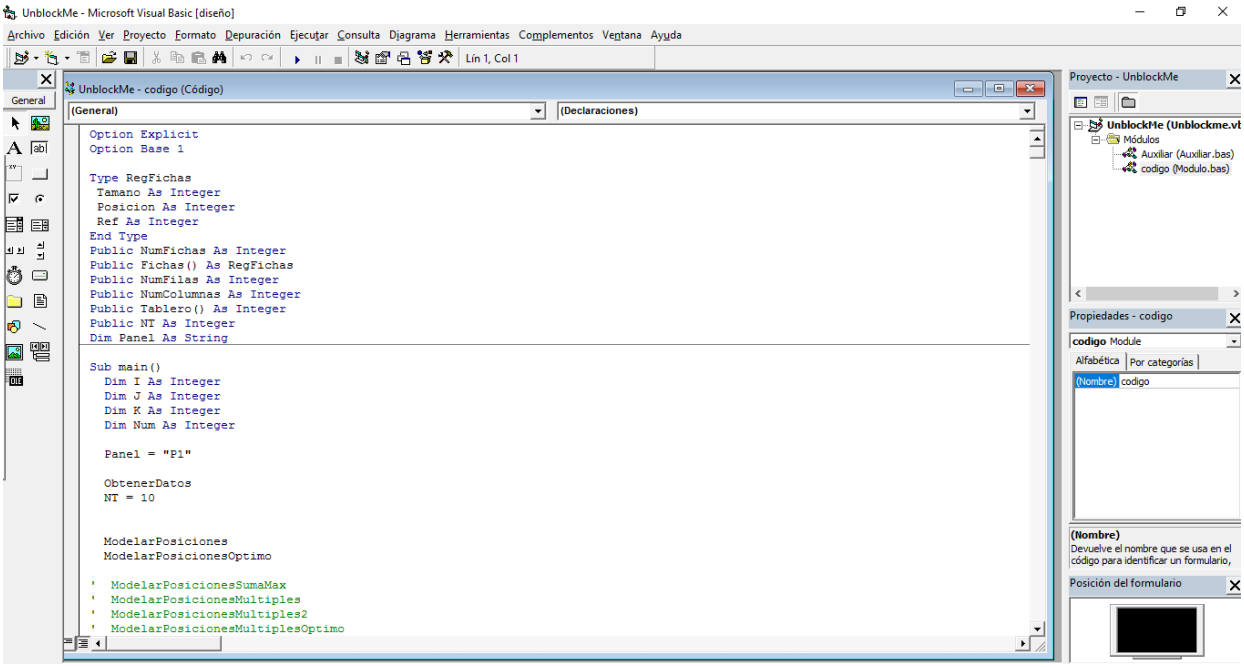

Imagen 4.3: código del proyecto seleccionado

Por último, para ejecutar el programa debemos pulsar el botón Play del menú superior y así conseguir que se generen los dos modelos utilizados para resolver mediante LINGO los paneles seleccionados. En la imagen 4.4 se muestra lo anteriormente descrito:

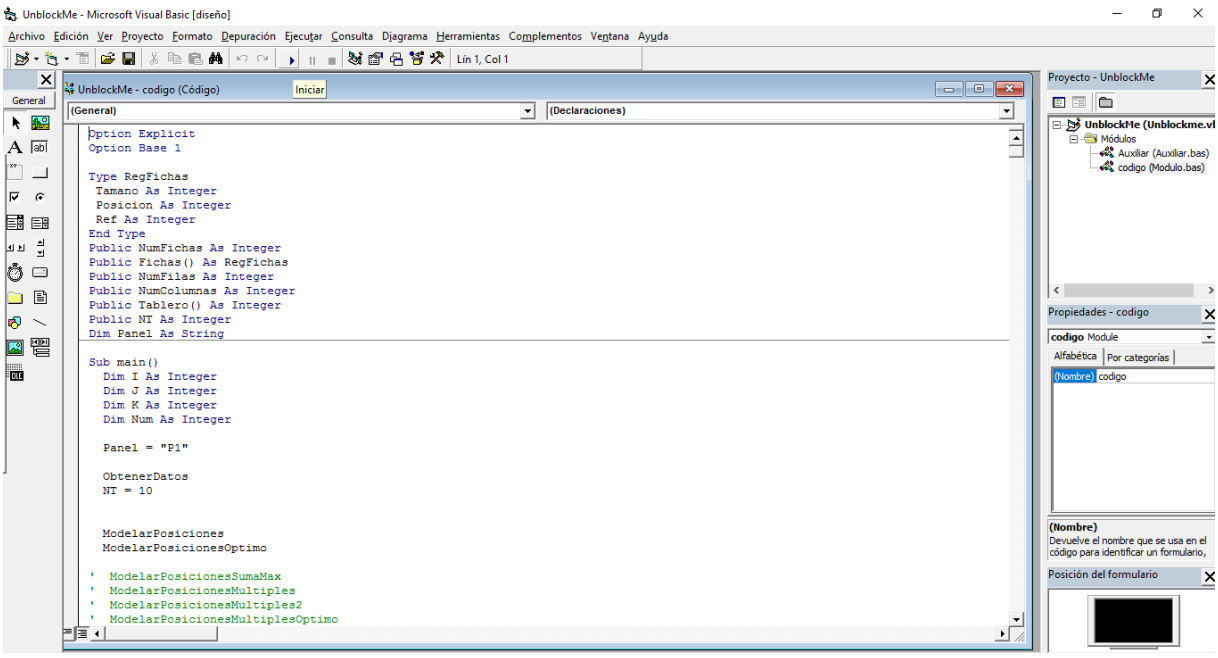

Imagen 4.4: botón con el que se ejecuta el programa Visual Basic

4.2. Herramienta de programación lineal: LINGO

En este trabajo fin de grado hemos empleado la herramienta de programación lineal LINGO. LINGO (LINear Generalize Optimizer) es una versátil herramienta para la formulación, resolución y análisis de problemas de programación lineal y no lineal. Esta herramienta nos proporciona un resultado que nos ayuda a encontrar la ganancia más alta o el costo más bajo.

Uno de los rasgos más destacables de esta herramienta es su lenguaje de modelado. Éste permite expresar un problema de una manera muy similar a la notación matemática normal. Además, es posible expresar sumas mediante expresiones iterativas, lo que hace que los modelos sean más compactos y fáciles de mantener.

Otro aspecto a destacar del lenguaje de modelado, es la posibilidad de poder leer datos de una hoja de cálculo, base de datos o, incluso, archivos de texto. Con esto, es más fácil hacer cambios en los datos de partida, y se reducen las posibilidades de cometer errores al plantear los modelos.

La estructura que sigue LINGO es la que se muestra en la imagen 4.5:

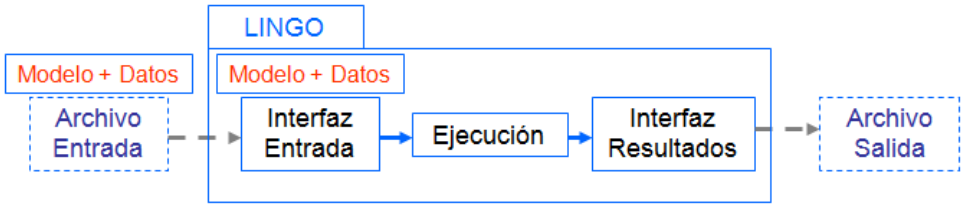

Imagen 4.5: estructura de la herramienta LINGO

En este trabajo fin de grado, tenemos un archivo de entrada, generado como se ha explicado anteriormente en Visual Basic. Luego, se abre con el programa LINGO y aparece el modelo con la estructura que hemos definido.

A continuación, ejecutamos el modelo para que LINGO encuentre la solución óptima al problema que le hemos planteado, la cual mostrará en un archivo de salida.

Los modelos de LINGO siguen el siguiente formato:

- Título: es la descripción del problema.
- Función objetivo: consiste en maximizar o minimizar la función que queremos optimizar.
- Restricciones del problema.
- Restricciones por las que se definen el tipo de variables que se emplean.

En cuanto a la sintaxis empleada por LINGO, ésta suele ser muy sencilla. Para el nombre de las variables y otros identificadores se pueden emplear hasta 32 caracteres, empezando con una letra y seguida de otras letras, dígitos o guión bajo. Además, no hace distinciones entre mayúsculas y minúsculas.

Respecto a las sentencias, todas deben terminar con un punto y coma. Para declarar la función objetivo, debemos escribir MAX o MIN seguidas del signo igual (=).

Por defecto, el valor de las variables es no negativo y continuo. Pero éstas pueden adoptar cualquier valor siempre y cuando se especifique. Para ello, LINGO proporciona cuatro funciones de dominio que permite sustituir el dominio predefinido. Estos dominios son:

- @GIN: restringe el dominio de la variable a valores enteros.
- @BIN: la variable toma valores binarios, es decir, 0 ó 1.
- @FREE: la variable puede asumir cualquier valor real.
- @BND: limita los valores a un rango finito.

Por último, al resolver el problema, muestra una ventana de resultados, en la que se nos especifica el número de iteraciones que ha necesitado para encontrar la solución óptima, y después nos informa del valor de la función objetivo en el punto óptimo. A continuación, nos muestra el valor de las variables básicas, así como su coste reducido. También hace referencia al sobrante y al precio sombra (dual price) de cada restricción.

Seguidamente, vamos a describir la estructura principal de la sintaxis de LINGO:

- Título: no es obligatorio, pero puede servirnos de ayuda para describir el modelo. Debe tener como máximo 128 caracteres y su formato se muestra en la imagen 4.6:

```
FORMATO:  {TITLE NOMBRE DEL MODELO ;}
```

Imagen 4.6: formato del título de los modelos en LINGO

- Función objetivo: para declararla, debemos colocar las palabras reservadas MIN o MAX seguidas del signo =. En la imagen 4.7 podemos ver su estructura:

```
FORMATO:  {[NOMBRE]} (MAX/MIN) = x1 + 2 * x2 + 3 * x3 – 5*x4;
```

Imagen 4.7: ejemplo de función objetivo

- Restricciones: LINGO permite nombrar las restricciones en sus modelos. Para ello, se inserta el nombre de la restricción entre corchetes delante de una línea de código. Su formato se muestra en la figura 4.8:

FORMATO: {[NOMBRE DE LA RESTRICCIÓN]} x1 + x2<=1;

Imagen 4.8: ejemplo de restricción

- Variables: a menos que se especifique lo contrario, el valor de las variables por defecto en LINGO son no negativas y continuas. Para establecer el valor de las variables, como hemos mencionado anteriormente, LINGO proporciona cuatro funciones de variables de dominio que sirven para sustituir el dominio que viene por defecto.

Para poder resolver los modelos en LINGO, lo primero que tenemos que hacer es abrir el programa y nos encontraremos con la siguiente pantalla, que se muestra en la imagen 4.9:

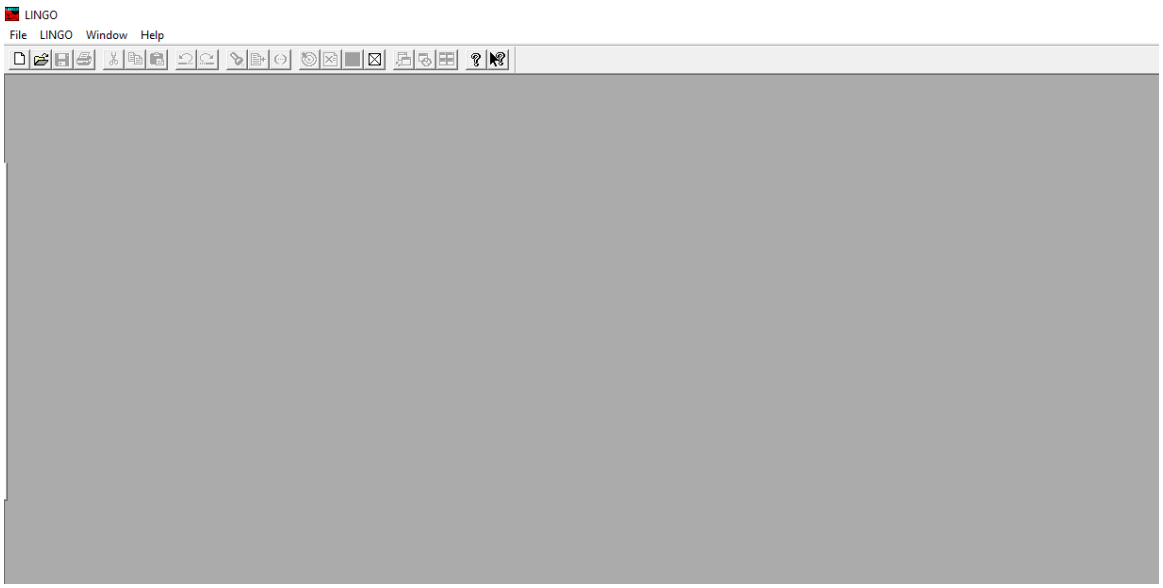

Imagen 4.9: pantalla principal de LINGO

Una vez que hemos abierto el programa, para abrir el modelo que queremos resolver tenemos que pulsar la pestaña File y, dentro de ésta, Open, tal y como se muestra en la imagen 4.10:

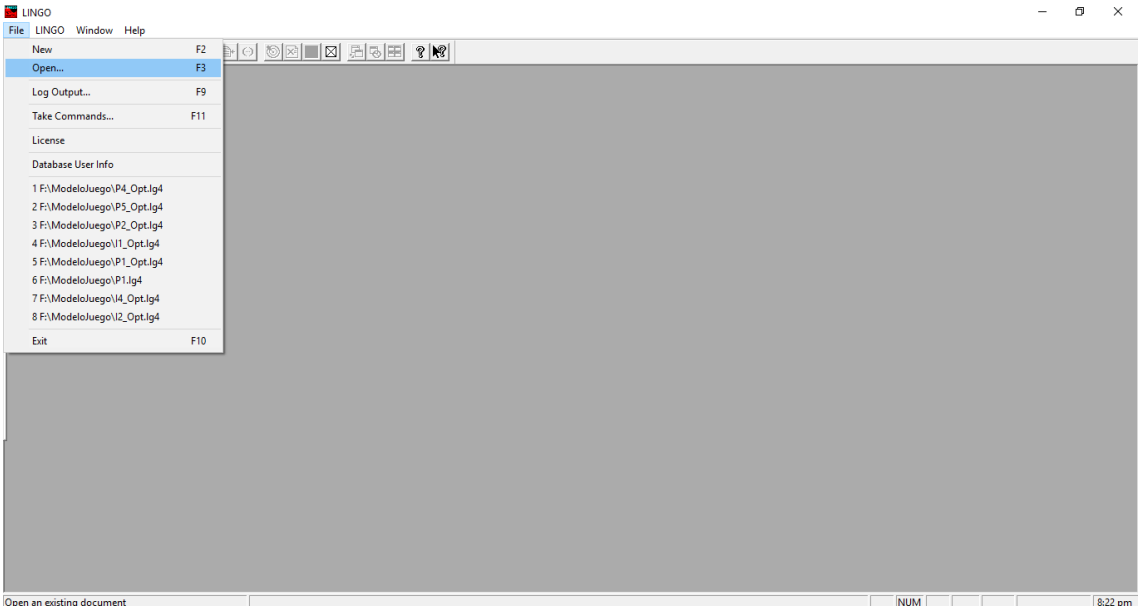

Imagen 4.10: abrir un modelo

Después, nos aparecerá una nueva ventana en la que debemos buscar el modelo que queremos resolver, el cual, al seleccionarlo, nos aparecerá en la ventana de LINGO, tal y como se observa en la imagen 4.11:

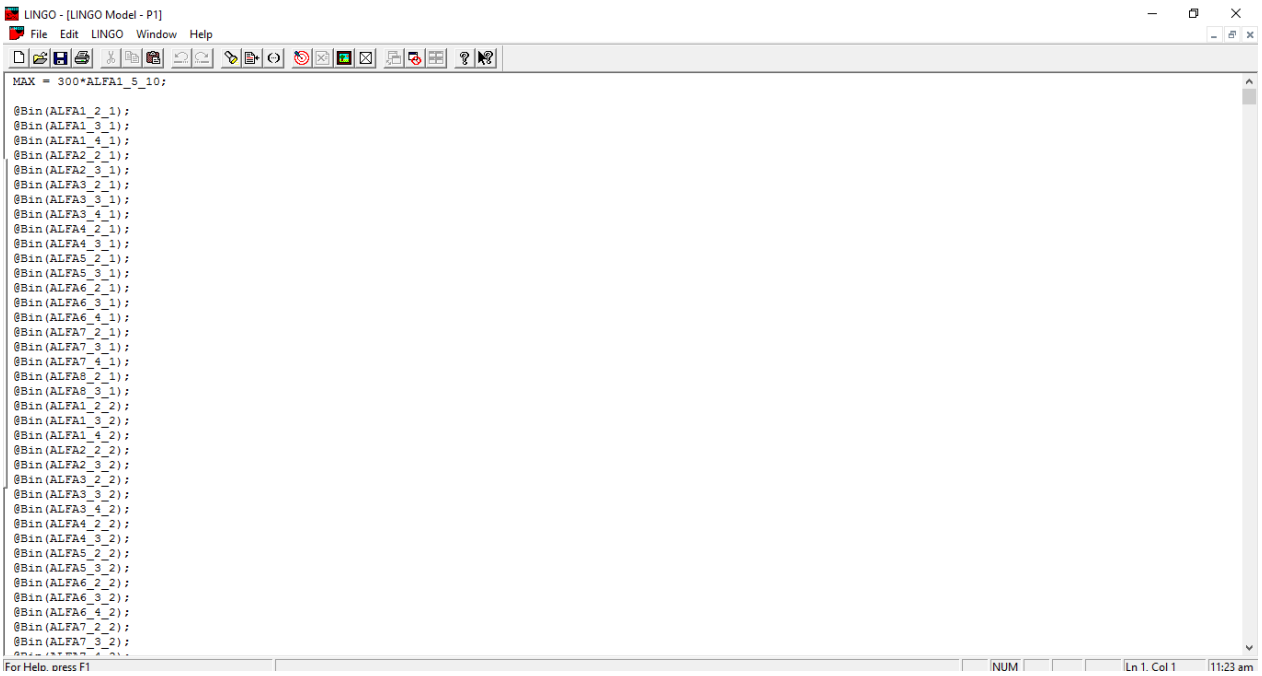

Imagen 4.11: ejemplo de modelo de LINGO

Por último, lo siguiente que habría que hacer es resolver el modelo. Para ello, hay que pulsar el botón de la parte superior del menú con forma de diana de color rojo, que se llama “Solve”. En la imagen 4.12 podemos ver el lugar donde se encuentra dicho botón:

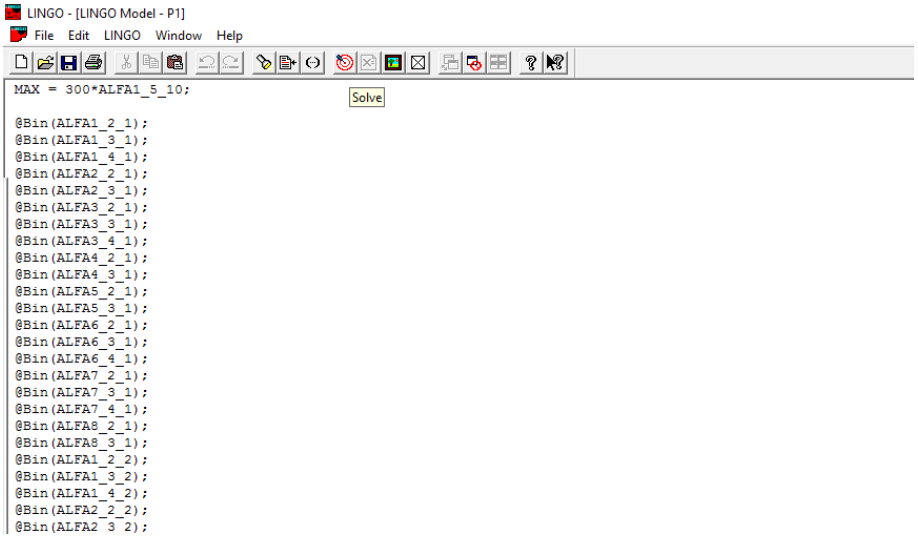

Imagen 4.12: icono “Solve” para resolver los modelos en LINGO

Una vez pulsado el botón de resolver, LINGO buscará el óptimo. Para ello, muestra una pantalla en la que aparecen datos sobre el valor que va tomando la función objetivo, así como el tiempo que lleva resolviendo el modelo. También podemos ver el número de variables que tiene el modelo, entre otros datos. En la imagen 4.13 observamos lo descrito anteriormente:

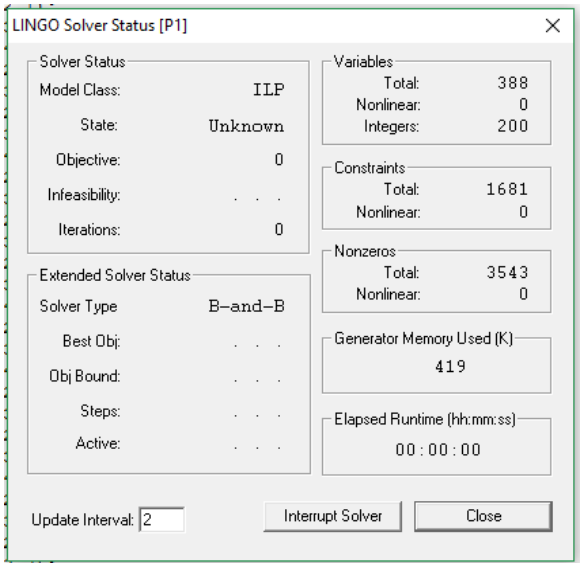

Imagen 4.13: pantalla secundaria “Solver” de LINGO

Cuando LINGO encuentra el óptimo, aparece una pantalla nueva en la que se muestra el valor de todas las variables, así como el de la función objetivo, tal y como se muestra en la imagen 4.14:

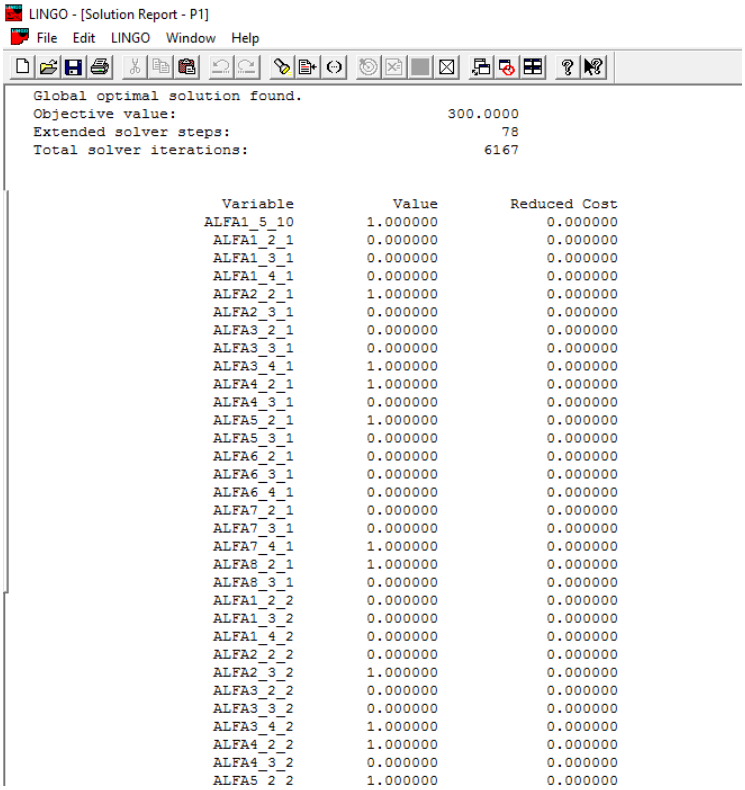

Imagen 4.14: pantalla de solución de un modelo en LINGO

5. RESULTADOS

En este apartado vamos a exponer los resultados obtenidos a partir del desarrollo de los modelos matemáticos, así como una comparativa entre las soluciones alcanzadas por dichos modelos y las obtenidas por personas.

En primer lugar, vamos a explicar cómo se han conseguido los datos aportados por los distintos puzles.

5.1. Paneles

Cada panel que queramos resolver tiene una estructura la cual vamos a explicar a continuación.

Lo primero que hay que hacer es enumerar las piezas que existan en cada panel con números que vayan desde el 1 hasta el número total de piezas. La primera pieza siempre es la de color rojo, siendo las demás el número que queramos.

Para este trabajo, hemos utilizado un documento ".txt" en el cual hemos ido introduciendo los datos necesarios para generar el problema a resolver.

El primer número que aparece en dicho documento es el número de piezas que presenta el panel. Debajo se muestran dos números que representan el tamaño del panel, los cuales siempre son de 6x6.

A continuación, se deja un espacio en blanco y justo después aparecen tantos números como piezas haya en el panel. En este caso, el número 1 hace referencia a las piezas que son horizontales y el número 2 a las piezas verticales.

De nuevo, se vuelve a dejar un espacio en blanco para, a continuación, describir el tamaño de cada pieza. En concreto, las piezas pueden tener un tamaño que ocupe 2 o 3 espacios. Por lo tanto, habrá tantos números como piezas haya en el puzle.

Ahora se vuelve a dejar un espacio en blanco, y nos encontramos con otro grupo de números (tantos como piezas haya) que hacen referencia a la fila o a la columna en la que se encuentra la primera parte de cada pieza.

Por último, podemos encontrar una matriz de tamaño 6x6 que representa numéricamente el puzle a resolver. Como se ha mencionado anteriormente, a cada pieza se le asigna un número, siempre y cuando la primera sea la de color rojo. Si no hay pieza en un lugar del panel, se escribe un 0, mientras que si existe pieza en ese sitio se pone el número de la pieza. Entre cada número se deja un espacio en blanco.

A continuación, se va a ejemplificar un archivo ".txt" correspondiente al puzle 1 de nivel principiante para poder entender con mayor facilidad el formato de los datos anteriormente explicados. Así, tal y como se observa en la imagen 5.1, el archivo quedaría de la siguiente forma:

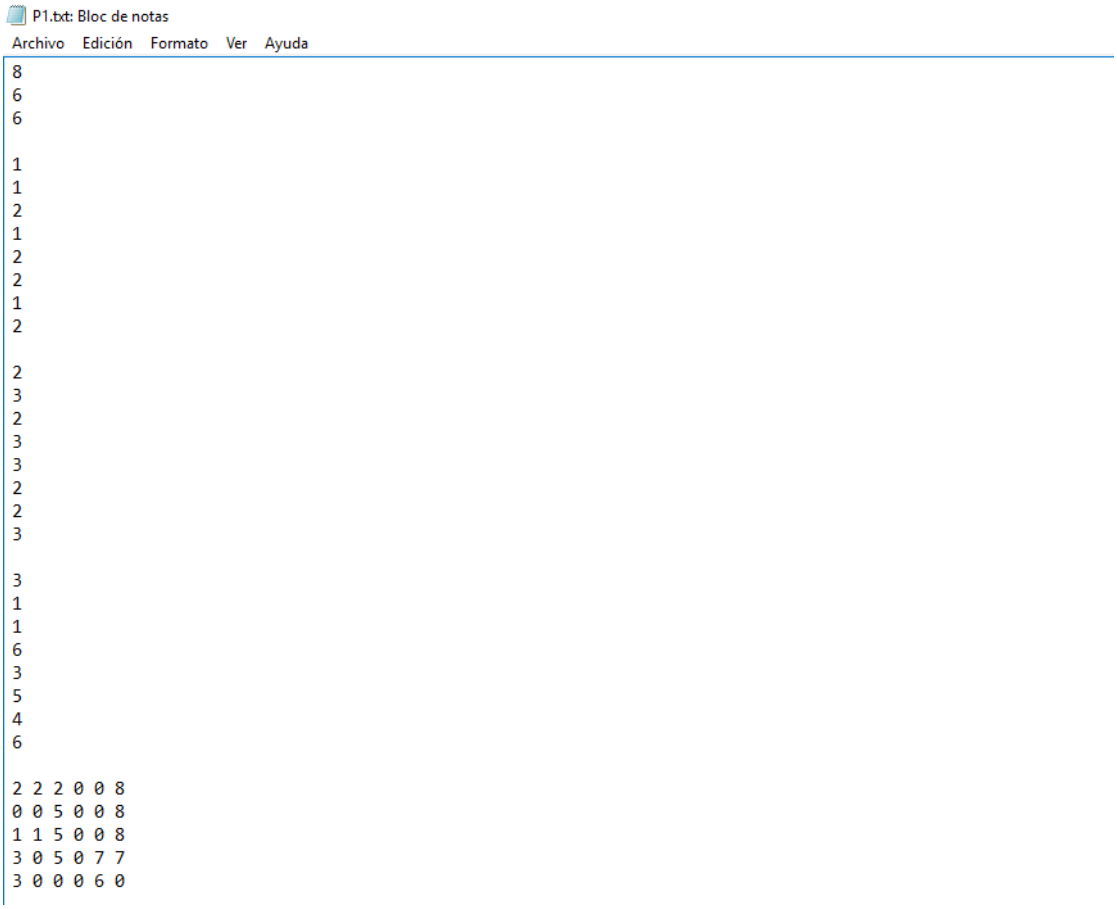

Imagen 5.1: ejemplo de archivo ".txt" donde se muestra el formato de los datos obtenidos para el puzle 1 de nivel principiante

En este caso, el número de piezas sería 8 y el número de filas y columnas, 6.

A continuación, encontramos 8 números, de los cuales 4 hacen referencia a piezas horizontales (cada número 1) y otros 4 a las verticales (cada número 2).

Después, encontramos otros 8 números, según el tamaño de cada pieza. En este caso, su tamaño puede ser 2 o 3.

El siguiente conjunto de 8 números hace referencia a la fila o columna donde se encuentra cada una de la primera parte de las piezas.

Por último, encontramos una matriz de 6x6 que representa la posición que ocupa cada pieza con el número que se le ha asignado y se rellena de 0 en las posiciones donde no hay ninguna pieza.

A continuación, se muestran las imágenes que corresponden a los 10 puzles empleados para desarrollar este trabajo.

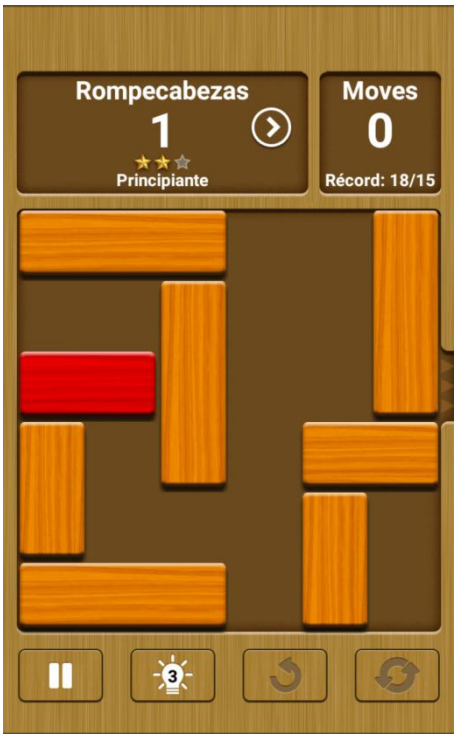

Imagen 5.2: puzle nivel principiante número 1

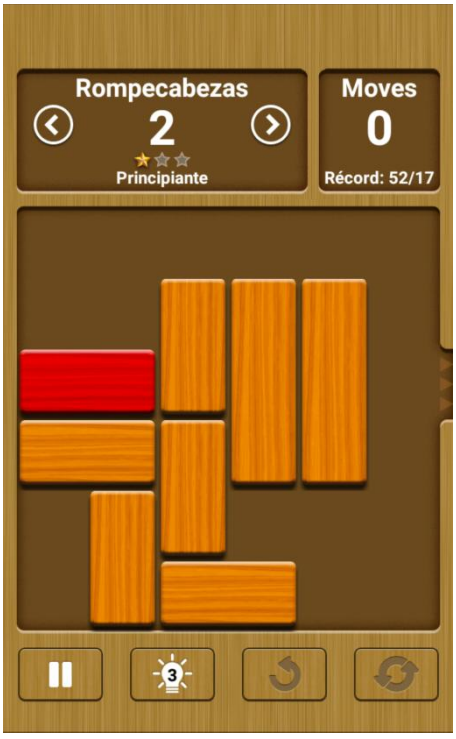

Imagen 5.3: puzle nivel principiante número 2

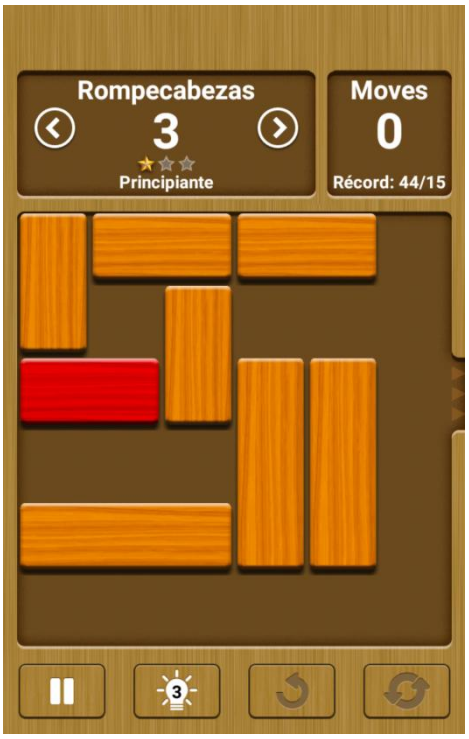

Imagen 5.4: puzle nivel principiante número 3

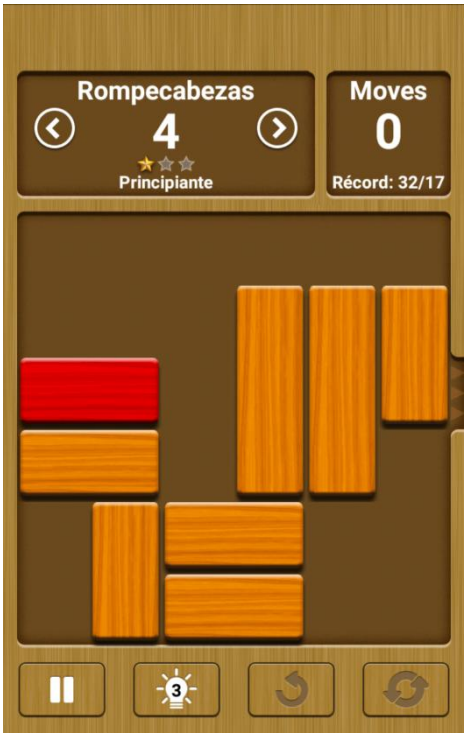

Imagen 5.5: puzle nivel principiante número 4

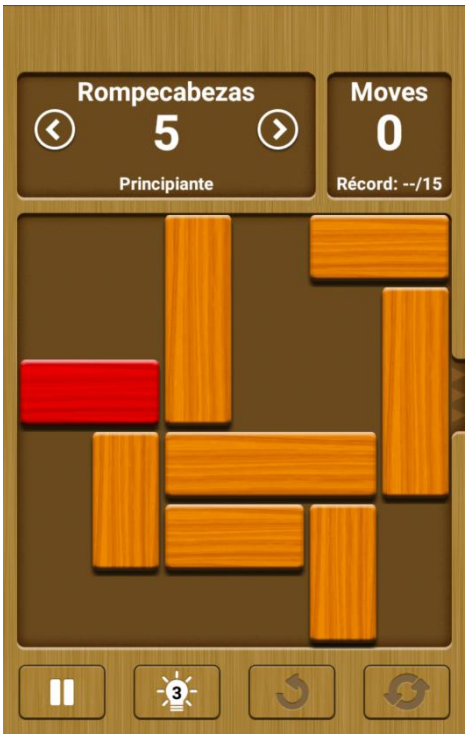

Imagen 5.6: puzle nivel principiante número 5

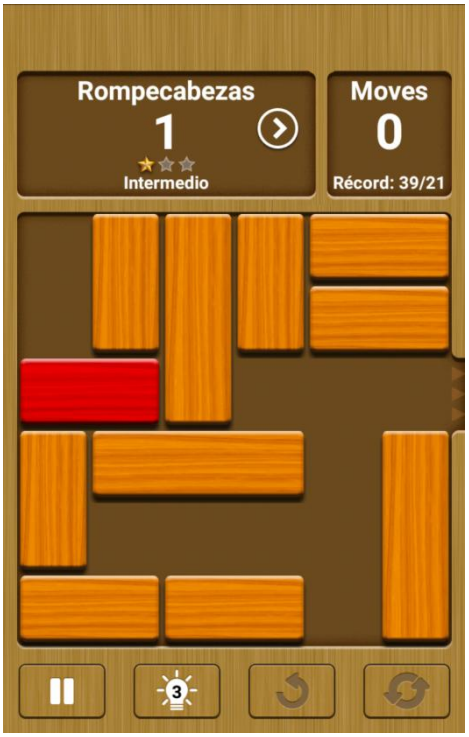

Imagen 5.7: puzle nivel intermedio número 1

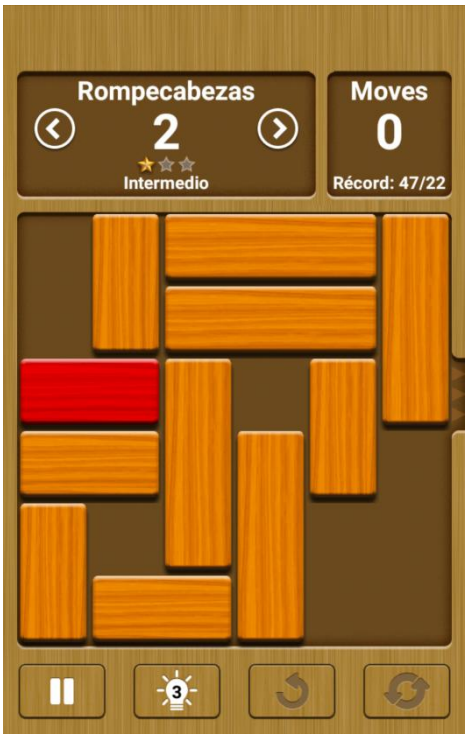

Imagen 5.8: puzle nivel intermedio número 2

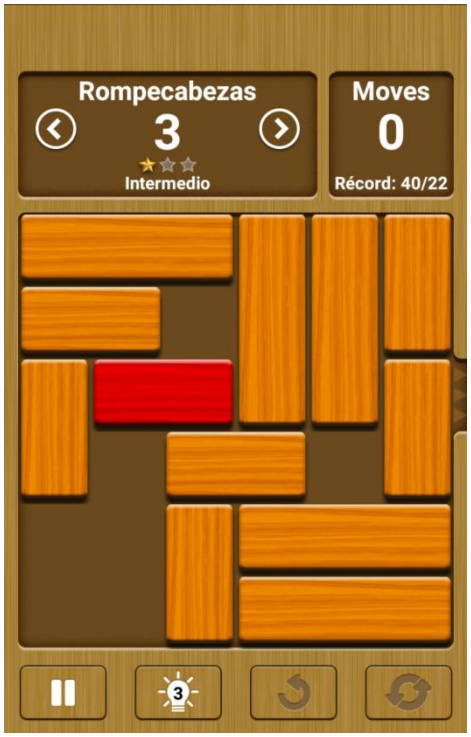

Imagen 5.9: puzle nivel intermedio número 3

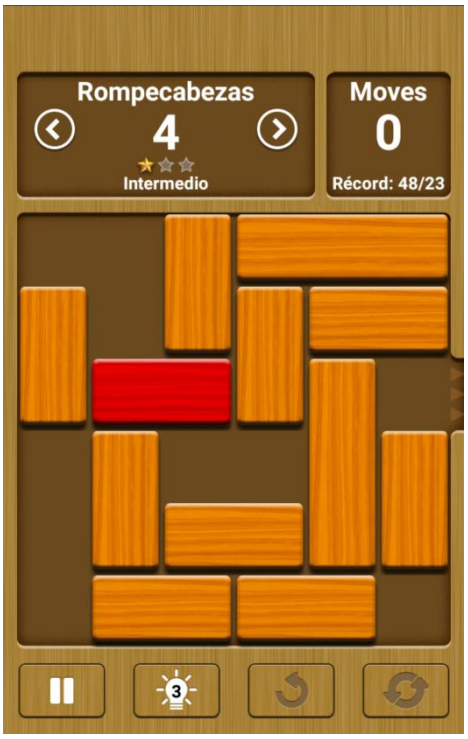

Imagen 5.10: puzle nivel intermedio número 4

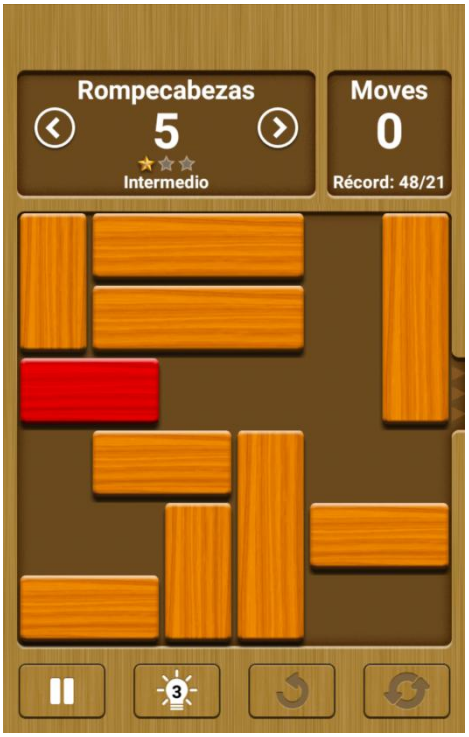

Imagen 5.11: puzle nivel intermedio número 5

5.2. Comparativa de tiempos y movimientos

En este subcapítulo, vamos a realizar una comparativa entre los tiempos que tardan los distintos modelos empleados en este trabajo y los de dos personas en solucionar cada uno de los paneles escogidos, así como otra tabla en la que se comparen los movimientos realizados en cada solución tanto por los modelos como por los jugadores.

Las tablas están compuestas por 5 columnas. En la primera de ellas, aparecen los distintos paneles que se han empleado, donde P e I hacen referencia al nivel de dificultad, en concreto principiante e intermedio, y los números del 1 al 5, los paneles que se han escogido de cada nivel.

En la siguiente columna aparece el modelo 1. Este modelo es el primero que empleamos, el cual es capaz de sacar la pieza pero necesita de un post-proceso para optimizar el número de pasos, que se explicará más adelante. En cada una de sus celdas aparece el tiempo que se ha requerido para que llegue a la solución óptima.

En la tercera columna nos encontramos con el modelo 2. Éste, aunque se ha diseñado como un modelo exacto, en la mayoría de los casos no ha sido capaz de llegar a la solución óptima después de más de 24 horas (en la tabla aparecen sus celdas con un asterisco), exceptuando P1 y P3.

En las dos últimas columnas aparecen los tiempos de dos personas que han intentado resolver cada uno de los paneles y a las que se les ha medido el tiempo que han tardado para ello. Jugador 1 y Jugador 2 son dos individuos que nunca antes habían probado el juego.

Tabla 5.1. Comparativa de tiempos de ambos modelos y dos jugadores.

	Modelo 1	Modelo 2	Jugador 1	Jugador 2
P1	0:02	2:28	2:42	1:58
P2	1:04	*	2:55	2:58
P3	0:01	1:16	1:44	1:48
P4	1:15	*	3:28	0:42
P5	0:55	*	1:41	1:51
I1	0:25	*	5:47	3:41
I2	0:06	*	3:21	1:49
I3	6:00	#	3:13	4:34
I4	2:59	#	5:34	4:30
I5	34:01	#	2:06	2:07

En la tabla, en el modelo 2, se observan varias celdas en las que encontramos los símbolos “*” y “#”. El primero hace referencia a que el modelo es capaz de llegar a una

solución admisible pero no es la óptima, después de un período de 24 horas. Sin embargo, el segundo símbolo significa que el modelo no ha sido capaz de llegar a una solución admisible en un período de 24 horas.

A continuación, se realiza un estudio mediante una tabla en la que se compara el número de movimientos realizados en cada modelo para los puzles y los que realizan los jugadores.

Tabla 5.2. Comparativa de movimientos entre los dos modelos y los dos jugadores.

	Modelo 1	Modelo 2	Jugador 1	Jugador 2
P1	20	25	55	43
P2	25	*	61	48
P3	19	9	39	34
P4	18	*	96	23
P5	18	*	31	39
I1	24	*	136	90
I2	25	*	78	48
I3	41	#	92	136
I4	45	#	180	160
I5	51	#	68	67

A continuación, se va a describir el post-proceso necesario para poder optimizar la solución que proporciona el modelo 1.

Como se ha mencionado anteriormente, este modelo, al no ser su función objetivo minimizar el número de pasos, puede realizar movimientos que no sean necesarios para llegar a la solución óptima ya que, al darle un número de períodos con suficiente margen y al sobrarle estados, el modelo, para completar dichos períodos, realiza este tipo de movimientos innecesarios. Además, es capaz de realizar un número de pasos mayor al necesario para poder mover una pieza hasta la posición a la que se desea llevar dicha pieza.

Por ello, una vez obtenida la solución a la que llega el modelo, se ha realizado un proceso de eliminación de estos movimientos no necesarios, el cual se ha llevado a cabo manualmente, realizando los movimientos que se obtienen de la solución de cada panel en el juego y eliminando los movimientos innecesarios y moviendo las piezas en un único paso si así se podía.

Seguidamente, se va a realizar una tabla en la que se muestre la solución a un mismo puzle mediante los dos modelos propuestos, en concreto el puzle de nivel principiante llamado anteriormente número 1.

Tabla 5.3. Solución al problema de nivel principiante 1, mediante el modelo 1.

ITERACIÓN 1	ITERACIÓN 2	ITERACIÓN 3	ITERACIÓN 4	ITERACIÓN 5
$\alpha_{1\_1\_1} = 1$	$\alpha_{1\_1\_2} = 1$	$\alpha_{1\_1\_3} = 1$	$\alpha_{1\_1\_4} = 1$	$\alpha_{1\_1\_5} = 1$
$\alpha_{2\_2\_1} = 1$	$\alpha_{2\_3\_2} = 1$	$\alpha_{2\_4\_3} = 1$	$\alpha_{2\_4\_4} = 1$	$\alpha_{2\_3\_5} = 1$
$\alpha_{3\_4\_1} = 1$	$\alpha_{3\_4\_2} = 1$	$\alpha_{3\_4\_3} = 1$	$\alpha_{3\_4\_4} = 1$	$\alpha_{3\_5\_5} = 1$
$\alpha_{4\_2\_1} = 1$	$\alpha_{4\_2\_2} = 1$	$\alpha_{4\_1\_3} = 1$	$\alpha_{4\_2\_4} = 1$	$\alpha_{4\_3\_5} = 1$
$\alpha_{5\_2\_1} = 1$	$\alpha_{5\_2\_2} = 1$	$\alpha_{5\_1\_3} = 1$	$\alpha_{5\_1\_4} = 1$	$\alpha_{5\_2\_5} = 1$
$\alpha_{6\_5\_1} = 1$	$\alpha_{6\_5\_2} = 1$	$\alpha_{6\_5\_3} = 1$	$\alpha_{6\_4\_4} = 1$	$\alpha_{6\_3\_5} = 1$
$\alpha_{7\_4\_1} = 1$	$\alpha_{7\_4\_2} = 1$	$\alpha_{7\_3\_3} = 1$	$\alpha_{7\_2\_4} = 1$	$\alpha_{7\_1\_5} = 1$
$\alpha_{8\_2\_1} = 1$	$\alpha_{8\_3\_2} = 1$	$\alpha_{8\_4\_3} = 1$	$\alpha_{8\_3\_4} = 1$	$\alpha_{8\_3\_5} = 1$
ITERACIÓN 6	ITERACIÓN 7	ITERACIÓN 8	ITERACIÓN 9	ITERACIÓN 10
$\alpha_{1\_1\_6} = 1$	$\alpha_{1\_2\_7} = 1$	$\alpha_{1\_3\_8} = 1$	$\alpha_{1\_4\_9} = 1$	$\alpha_{1\_5\_10} = 1$
$\alpha_{2\_2\_6} = 1$	$\alpha_{2\_2\_7} = 1$	$\alpha_{2\_2\_8} = 1$	$\alpha_{2\_2\_9} = 1$	$\alpha_{2\_2\_10} = 1$
$\alpha_{3\_5\_6} = 1$	$\alpha_{3\_5\_7} = 1$	$\alpha_{3\_5\_8} = 1$	$\alpha_{3\_5\_9} = 1$	$\alpha_{3\_5\_10} = 1$
$\alpha_{4\_3\_6} = 1$	$\alpha_{4\_4\_7} = 1$	$\alpha_{4\_4\_8} = 1$	$\alpha_{4\_3\_9} = 1$	$\alpha_{4\_2\_10} = 1$
$\alpha_{5\_3\_6} = 1$	$\alpha_{5\_4\_7} = 1$	$\alpha_{5\_4\_8} = 1$	$\alpha_{5\_3\_9} = 1$	$\alpha_{5\_3\_10} = 1$
$\alpha_{6\_3\_6} = 1$	$\alpha_{6\_2\_7} = 1$	$\alpha_{6\_2\_8} = 1$	$\alpha_{6\_1\_9} = 1$	$\alpha_{6\_1\_10} = 1$
$\alpha_{7\_1\_6} = 1$	$\alpha_{7\_1\_7} = 1$	$\alpha_{7\_1\_8} = 1$	$\alpha_{7\_1\_9} = 1$	$\alpha_{7\_1\_10} = 1$
$\alpha_{8\_3\_6} = 1$	$\alpha_{8\_2\_7} = 1$	$\alpha_{8\_3\_8} = 1$	$\alpha_{8\_4\_9} = 1$	$\alpha_{8\_4\_10} = 1$

Para poder entender el post-proceso anteriormente descrito, se van a poner una serie de ejemplos a continuación:

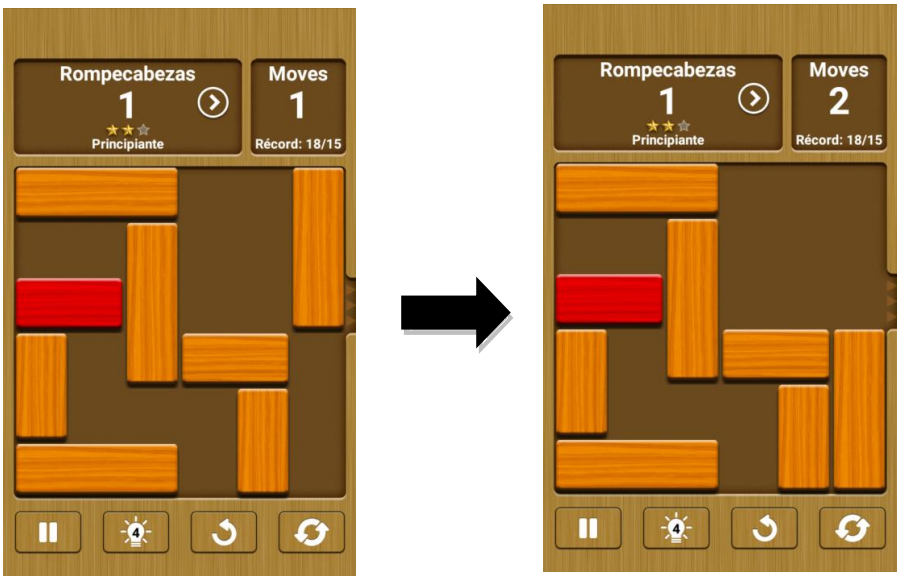

Imagen 5.12: ejemplo de movimiento que el modelo 1 contaría como varios pasos. Como podemos ver en esta imagen, la pieza vertical de tamaño 3 situada a la derecha del panel, que en el modelo se le ha asignado el número 8, después de tres iteraciones, se situaría en la posición en la que aparece en la imagen de la derecha. Sin embargo, estos movimientos se podrían realizar en uno solo, disminuyendo así el número de pasos totales empleados para solucionar el problema.

En la imagen 5.13, podemos ver otro ejemplo como el anterior, en el que la pieza numerada como 4 se desplaza a la izquierda en varios pasos, desde la iteración 8 a la 10.

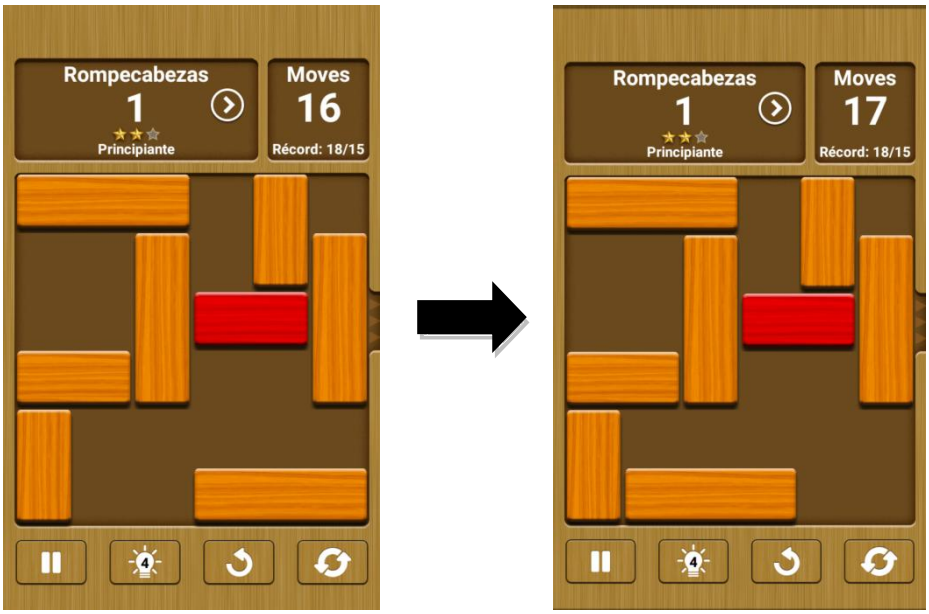

Imagen 5.13: ejemplo de cómo el modelo 1 realiza un posible movimiento en varios pasos.

En la imagen 5.14 podemos observar cómo el modelo 1 es capaz de añadir movimientos innecesarios una vez obtenida ya la solución óptima.

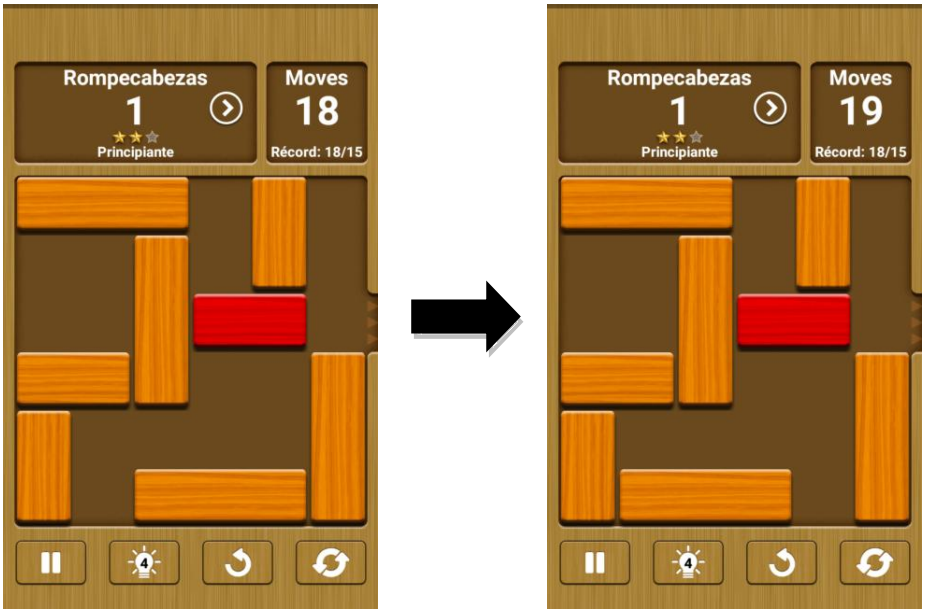

Imagen 5.14: en la iteración 10, se puede apreciar cómo la variable correspondiente a la pieza número 4 cambia de índice para su posición, es decir, que la pieza cambia de la posición 3 en la iteración 9, a la 2, en la siguiente, sin influir en el camino ya despejado de la pieza objetivo, por lo que añade un movimiento más a la solución innecesario.

Tabla 5.4. Solución al problema de nivel principiante 1, mediante el modelo 2.

ITERACIÓN 1		ITERACIÓN 2	
$\alpha_{1\_1\_1} = 1$	$MOVE_{1\_1\_1} = 0$	$\alpha_{1\_1\_2} = 1$	$MOVE_{1\_1\_2} = 0$
$\alpha_{2\_2\_1} = 1$	$MOVE_{2\_1\_1} = 1$	$\alpha_{2\_3\_2} = 1$	$MOVE_{2\_2\_2} = 1$
$\alpha_{3\_4\_1} = 1$	$MOVE_{3\_4\_1} = 0$	$\alpha_{3\_4\_2} = 1$	$MOVE_{3\_4\_2} = 0$
$\alpha_{4\_1\_1} = 1$	$MOVE_{4\_1\_1} = 0$	$\alpha_{4\_1\_2} = 1$	$MOVE_{4\_1\_2} = 0$
$\alpha_{5\_2\_1} = 1$	$MOVE_{5\_2\_1} = 0$	$\alpha_{5\_2\_2} = 1$	$MOVE_{5\_2\_2} = 0$
$\alpha_{6\_5\_1} = 1$	$MOVE_{6\_5\_1} = 0$	$\alpha_{6\_5\_2} = 1$	$MOVE_{6\_5\_2} = 0$
$\alpha_{7\_5\_1} = 1$	$MOVE_{7\_5\_1} = 0$	$\alpha_{7\_4\_2} = 1$	$MOVE_{7\_5\_2} = 1$
$\alpha_{8\_1\_1} = 1$	$MOVE_{8\_1\_1} = 0$	$\alpha_{8\_1\_2} = 1$	$MOVE_{8\_1\_2} = 0$
ITERACIÓN 3		ITERACIÓN 4	
$\alpha_{1\_1\_3} = 1$	$MOVE_{1\_1\_3} = 0$	$\alpha_{1\_1\_4} = 1$	$MOVE_{1\_1\_4} = 0$
$\alpha_{2\_4\_3} = 1$	$MOVE_{2\_3\_3} = 1$	$\alpha_{2\_4\_4} = 1$	$MOVE_{2\_4\_4} = 0$
$\alpha_{3\_4\_3} = 1$	$MOVE_{3\_4\_3} = 0$	$\alpha_{3\_5\_4} = 1$	$MOVE_{3\_4\_4} = 1$
$\alpha_{4\_1\_3} = 1$	$MOVE_{4\_1\_3} = 0$	$\alpha_{4\_2\_4} = 1$	$MOVE_{4\_1\_4} = 1$
$\alpha_{5\_1\_3} = 1$	$MOVE_{5\_2\_3} = 1$	$\alpha_{5\_1\_4} = 1$	$MOVE_{5\_1\_4} = 0$
$\alpha_{6\_5\_3} = 1$	$MOVE_{6\_5\_3} = 0$	$\alpha_{6\_5\_4} = 1$	$MOVE_{6\_5\_4} = 0$
$\alpha_{7\_3\_3} = 1$	$MOVE_{7\_4\_3} = 1$	$\alpha_{7\_2\_4} = 1$	$MOVE_{7\_3\_4} = 1$
$\alpha_{8\_2\_3} = 1$	$MOVE_{8\_1\_3} = 1$	$\alpha_{8\_2\_4} = 1$	$MOVE_{8\_2\_4} = 0$
ITERACIÓN 5		ITERACIÓN 6	
$\alpha_{1\_1\_5} = 1$	$MOVE_{1\_1\_5} = 0$	$\alpha_{1\_1\_6} = 1$	$MOVE_{1\_1\_6} = 0$
$\alpha_{2\_4\_5} = 1$	$MOVE_{2\_4\_5} = 0$	$\alpha_{2\_4\_6} = 1$	$MOVE_{2\_4\_6} = 0$
$\alpha_{3\_5\_5} = 1$	$MOVE_{3\_5\_5} = 0$	$\alpha_{3\_5\_6} = 1$	$MOVE_{3\_5\_6} = 0$
$\alpha_{4\_2\_5} = 1$	$MOVE_{4\_2\_5} = 0$	$\alpha_{4\_3\_6} = 1$	$MOVE_{4\_2\_6} = 1$
$\alpha_{5\_2\_5} = 1$	$MOVE_{5\_1\_5} = 1$	$\alpha_{5\_3\_6} = 1$	$MOVE_{5\_2\_6} = 1$
$\alpha_{6\_5\_5} = 1$	$MOVE_{6\_5\_5} = 0$	$\alpha_{6\_4\_6} = 1$	$MOVE_{6\_5\_6} = 1$
$\alpha_{7\_1\_5} = 1$	$MOVE_{7\_2\_5} = 1$	$\alpha_{7\_1\_6} = 1$	$MOVE_{7\_1\_6} = 0$
$\alpha_{8\_2\_5} = 1$	$MOVE_{8\_2\_5} = 0$	$\alpha_{8\_3\_6} = 1$	$MOVE_{8\_2\_6} = 1$
ITERACIÓN 7		ITERACIÓN 8	
$\alpha_{1\_2\_7} = 1$	$MOVE_{1\_1\_7} = 1$	$\alpha_{1\_3\_8} = 1$	$MOVE_{1\_2\_8} = 1$
$\alpha_{2\_4\_7} = 1$	$MOVE_{2\_4\_7} = 0$	$\alpha_{2\_4\_8} = 1$	$MOVE_{2\_4\_8} = 0$
$\alpha_{3\_5\_7} = 1$	$MOVE_{3\_5\_7} = 0$	$\alpha_{3\_5\_8} = 1$	$MOVE_{3\_5\_8} = 0$
$\alpha_{4\_4\_7} = 1$	$MOVE_{4\_3\_7} = 1$	$\alpha_{4\_4\_8} = 1$	$MOVE_{4\_4\_8} = 0$
$\alpha_{5\_4\_7} = 1$	$MOVE_{5\_3\_7} = 1$	$\alpha_{5\_4\_8} = 1$	$MOVE_{5\_4\_8} = 0$
$\alpha_{6\_4\_7} = 1$	$MOVE_{6\_4\_7} = 0$	$\alpha_{6\_4\_8} = 1$	$MOVE_{6\_4\_8} = 0$
$\alpha_{7\_1\_7} = 1$	$MOVE_{7\_1\_7} = 0$	$\alpha_{7\_1\_8} = 1$	$MOVE_{7\_1\_8} = 0$
$\alpha_{8\_3\_7} = 1$	$MOVE_{8\_3\_7} = 0$	$\alpha_{8\_3\_8} = 1$	$MOVE_{8\_3\_8} = 0$
ITERACIÓN 9		ITERACIÓN 10	
$\alpha_{1\_4\_9} = 1$	$MOVE_{1\_3\_9} = 1$	$\alpha_{1\_5\_10} = 1$	$MOVE_{1\_4\_10} = 1$
$\alpha_{2\_4\_9} = 1$	$MOVE_{2\_4\_9} = 0$	$\alpha_{2\_4\_10} = 1$	$MOVE_{2\_4\_10} = 0$
$\alpha_{3\_5\_9} = 1$	$MOVE_{3\_5\_9} = 0$	$\alpha_{3\_5\_10} = 1$	$MOVE_{3\_5\_10} = 0$
$\alpha_{4\_4\_9} = 1$	$MOVE_{4\_4\_9} = 0$	$\alpha_{4\_3\_10} = 1$	$MOVE_{4\_4\_10} = 1$
$\alpha_{5\_3\_9} = 1$	$MOVE_{5\_4\_9} = 1$	$\alpha_{5\_3\_10} = 1$	$MOVE_{5\_3\_10} = 0$
$\alpha_{6\_4\_9} = 1$	$MOVE_{6\_4\_9} = 0$	$\alpha_{6\_4\_10} = 1$	$MOVE_{6\_4\_10} = 0$
$\alpha_{7\_1\_9} = 1$	$MOVE_{7\_1\_9} = 0$	$\alpha_{7\_1\_10} = 1$	$MOVE_{7\_1\_10} = 0$
$\alpha_{8\_3\_9} = 1$	$MOVE_{8\_3\_9} = 0$	$\alpha_{8\_4\_10} = 1$	$MOVE_{8\_3\_10} = 1$

6. CONCLUSIONES

En este trabajo fin de grado se ha formulado matemáticamente un problema de optimización asociado a un juego de ordenador, llamado *Unblock Me*, cuya formulación ha sido bastante compleja, basada en períodos que son indeterminados, ya que no se conoce cuando se va a llegar a la resolución de cada puzle. Sin embargo, se ha conseguido resolver algunos problemas de forma exacta utilizando programación en LINGO. Por tanto, se puede decir que este problema se puede resolver mediante programación matemática dentro de unos límites (tiempo, movimientos, variables, etc.).

Una vez que se han mostrado los resultados del trabajo, se van a desarrollar una serie de conclusiones respecto al mismo. En general, el resultado final ha sido bastante satisfactorio, a pesar de la cantidad de problemas que han ido saliendo tanto en la programación como en la resolución del problema que se ha desarrollado.

En primer lugar, destacar haber trabajado con cierta profundidad herramientas para la resolución de problemas de programación matemática, como son las herramientas LINGO y Visual Basic. Además, me ha servido desarrollar este proyecto para poner en práctica tanto conocimientos como técnicas que he adquirido durante el grado.

La implementación utilizada ha sido muy compleja, debido a la cantidad de índices empleados. Aún así, se ha conseguido realizar un modelo mediante Visual Basic bastante bueno, ya que ha sido capaz de encontrar solución a la mayoría de puzles empleados en este trabajo.

Otro dato a tener en cuenta ha sido la comparativa de tiempos que se ha empleado tanto por parte de los modelos como por parte de los jugadores. De esto, se puede deducir que el modelo 2 se asemeja más a la solución a la que llegan los jugadores, ya que las piezas se pueden mover varias posiciones en el mismo período. Además, podemos observar que el modelo 1 cuando se avanza de nivel va encontrando más dificultades a la hora de encontrar una solución óptima.

Por otro lado, en cuanto al número de movimientos, el modelo 1 se asemeja al número de movimientos que realizan los jugadores, ya que del segundo modelo apenas se tienen datos para poder comparar. Aún así, tal y como se ha mencionado anteriormente, el modelo 1, a la par que avanza la dificultad de los puzles, empieza a encontrar dificultad para hallar una solución óptima.

Se ha realizado un primer modelo el cual busca llevar la pieza objetivo a la salida del puzle sin importar repetir movimientos. Sin embargo, se han ido realizando alternativas al primer modelo hasta encontrar uno que, además de llevar la pieza a la salida, era capaz de reducir el número de movimientos. No obstante, cuando se busca

optimalidad, el modelo se vuelve infinitamente más lento, por lo que se podría buscar algunos procedimientos mediante los cuales se pueda alcanzar la optimalidad.

Como conclusión general, se puede afirmar que la programación matemática puede ser útil para determinados problemas que no superen una cierta complejidad o un cierto nivel en el juego estudiado. Aunque, cuando la velocidad de los ordenadores aumente, quizás el rango de problemas que se pueden solucionar mediante la programación matemática también se amplíe.

7. BIBLIOGRAFÍA

- Descripción del juego *Unblock Me Free*: <https://www.amazon.es/Kiragames-Co-Ltd-Unblock-FREE/dp/B00HIEMXLK>

- Descripción del juego *Unblock Me Free*: <https://play.google.com/store/apps/details?id=com.kiragames.unblockmefree&hl=es 419>

- Visual Basic.
  Autor: José Eduardo Maluf de Carvalho.
  Editorial: McGraw-Hill.

- Enciclopedia de Visual Basic.
  Autor: Francisco Javier Ceballos.
  Editorial: ra-ma.

- Introducción a LINGO:
  <https://bibing.us.es/proyectos/abreproy/70427/fichero/anexos.pdf>

- Investigación Operativa 2002, Software Para Programación Lineal LINGO.
  Autores: Erica Canizo, Paola Lucero.