Propuesta de trabajo "Buscaminas" con redes bayesianas – Inteligencia Artificial 2017/18

Mª Victoria Calbet González
Universidad de Sevilla
Sevilla, España
marcalgon4@alum.us.es – victoria_calbet@hotmail.com

Eva Ramos Castro
Universidad de Sevilla
Sevilla, España
evaramcas1@alum.us.es - evitaramos43@gmail.com

***Resumen*— Implementar un método que determine las casillas con mayor probabilidad de no contener una mina y sugerirlas como el siguiente paso a seleccionar.**

Palabras Clave*— *Tablero, casilla, matriz, fila, columna, mina, probabilidad, interfaz, bayesiana, num_vecinos, sum_vecinos, nodo, colindantes, vecinos.

I. Introducción

El buscaminas es un videojuego para un jugador, inventado por Robert Donner en 1989. El objetivo del juego es despejar un campo de minas sin detonar ninguna. En el juego se parte de una cuadrícula de dimensiones conocidas en la que hay escondidas una cantidad predeterminada de minas. En cada jugada, se descubre una de las casillas, si la casilla descubierta contiene una mina, se da por perdido el juego, en cambio, si la casilla descubierta no contiene una mina, permite despejar una zona de la cuadrícula hasta las casillas que limitan con alguna casilla que contenga una mina, apareciendo información referente al número de minas en las casillas colindantes.

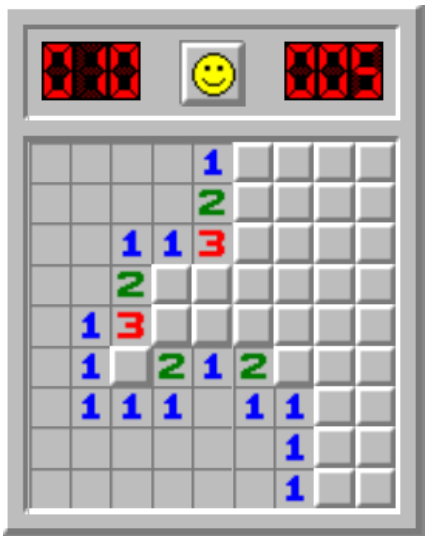


Ilustración 1. Partida de buscaminas

El objetivo principal del problema propuesto es la implementación de un método que determine las casillas con mayor probabilidad de no contener una mina y sugerirlas como siguiente paso a dar en el juego. Para ello se debe describir la situación actual del juego mediante una red bayesiana de forma similar a la descrita en [2][2], asociada a un tablero dado, siendo necesaria esta información para la creación de la red.

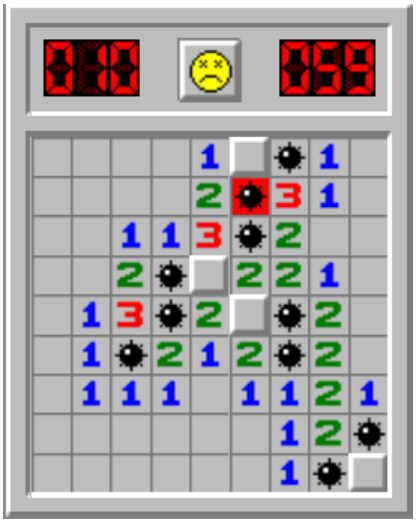


Ilustración 2. Partida perdida del buscaminas.

Con este objetivo, lo que se pretende es conseguir ganar una partida sin caer en ninguna casilla con minas y perder como se observa en la Ilustración 2.

II. Objetivo

Se ha implementado un método que se encargue de construir la red bayesiana a partir de un tablero que depende del número de filas y columnas, así como el número de minas que este contenga. A su vez, también se ha realizado el desarrollo de un mecanismo para la obtención de la probabilidad y la sugerencia de la casilla más óptima para dar el paso siguiente en el juego, es decir, la casilla con la probabilidad más baja de obtener una mina; para ello se ha empleado un sistema de eliminación de variables, en función de las casillas que aún no están descubiertas.

III. Otras consideraciones

Tenemos en conocimiento, que no todas las personas que juegan al buscaminas lo hacen siguiendo una lógica o fijándose con respecto a las casillas colindantes y la información que presentan, e incluso jugadores que tienen en cuenta estos datos a la hora de clicar en una casilla, pueden fallar. Tras leer y pensar en posibles soluciones con las cuáles podríamos abordar la solución del problema, hemos optado por una solución teniendo en cuenta la participación del usuario, de esta forma, sería una buena oportunidad que la resolución del problema esté implementada en una interfaz gráfica para que un usuario que se disponga a querer jugar al buscaminas, la ejecute y le sirva como apoyo para resolver el tablero.

Una vez puestos en preámbulo, explicaremos con mayor detalle todo el proceso que se ha realizado para la resolución del problema, así como los inconvenientes encontrados y las soluciones propuestas, en los siguientes apartados.

IV. Metodología

A. *Métodos empleados*

Para la resolución del problema se ha empleado un sistema basado en redes bayesianas, que para la resolución de la probabilidad se ha utilizado un sistema de inferencia exacta sobre el nodo o casilla sin descubrir (eliminación de variables).

Para ello se realizarán los siguientes pasos:

- Obtención del grafo con minas. [Pseudocódigo a]
- Método que relaciona las aristas entre los nodos, tanto para número de vecinos como para la suma de la información de los vecinos.
- Crear el modelo.
- Crear distribución de probabilidad condicional.
- Método para crear la probabilidad con respecto a la suma de minas adyacentes a una casilla sin descubrir.
- Método de eliminación de variables.
- Método que obtiene la mejor probabilidad de no tener mina de una casilla sin descubrir.

V. del papel a la implementación

A la hora de plantearnos la solución de nuestro problema, la primera toma de decisión ha sido la elaboración de un tablero, una matriz, que dependerá del número de filas y columnas, que tendrá tantas como queramos que sea nuestro tablero. Para ello, consideraremos que, si una matriz tiene n filas y m columnas, irá desde 0 hasta n-1 y desde 0 hasta m-1.

$$matriz = n \times m$$

$$donde\ matriz = \begin{pmatrix} (0,0) & \cdots & (0,m-1) \\ \vdots & \ddots & \vdots \\ (n-1,0) & \cdots & (n-1,m-1) \end{pmatrix}$$

```
Obtener grafo con minas
Entradas: filas, columnas, minas
Salidas: grafo
Definir minas, grafo como list
Definir x como false
Definir nmin como número de minas
    Para i hasta filas Hacer
        Escribir [] ← grafo
            Para j hasta columnas:
                Escribir grafo[i] ← x
            FinPara
    FinPara
    Mientras nmin < 0 Hacer
        Para p←0 hasta num_minas hacer
            a ← random (0, filas-1)
            b ← random (0, columnas-1)
            var ← (a,b)
            Si var ~ minas Entonces
              Si nmin > 0 Entonces
                nmin ← nmin-1
                Escribir minas[(a,b)]
                Para i hasta grafo hacer
                    Para j hasta grafo[i] hacer
                      Si i==a & j==b Entonces
                        x ← true
                        Escribir grafo[i][j] ← (x)
                      FinSi
                    FinPara
                FinPara
```

Pseudocódigo a. Obtener grafo con minas

El tablero no consideraremos que será siempre de tamaño cuadrado, pues creemos, que un tablero de buscaminas puede tener un tamaño de filas distinto al número de minas.

A partir de la obtención de cómo será nuestro tablero, nos planteamos las siguientes cuestiones, como, *¿Qué variables dependen para que exista una mina en una casilla? ¿En qué situaciones una casilla contiene una mina de manera inequívoca?*

Como respuesta a estas preguntas llegamos a la conclusión, que una variable fundamental para la resolución del problema era el número de minas y las casillas vecinas de los nodos, puesto que una casilla tendrá más probabilidad de contener una mina cuanto mayor sea la suma de la información contenida en las casillas colindantes y a su vez, dependiendo del número de vecinos con respecto al nodo o casilla en la cual se encuentre, es decir:

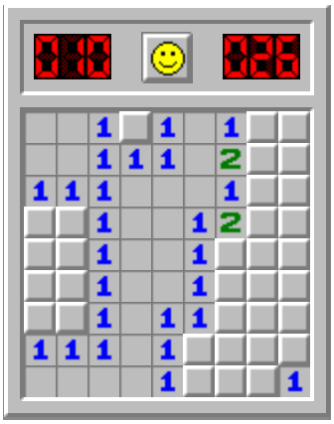


Ilustración 3. Ejemplo buscaminas para explicar la probabilidad

Teniendo en cuenta el ejemplo de la Ilustración 3 y la explicación de cómo se generará el tamaño de la matriz, tenemos un tablero con filas n = 9, columnas n = 9 y minas = 10; la casilla o nodo (0,3) tiene 5 casillas colindantes o evidencias, y la casilla (0,7) tendrá 2 casillas colindantes o evidencias. Si tenemos en cuenta estos datos solamente, la posición de la casilla sin descubrir y las casillas colindantes descubiertas, podremos obtener una probabilidad de obtención de mina, pero no será lo más eficiente posible, pues al depender solamente del número de casillas colindantes, siempre nos considerará una casilla sin evidencias a su alrededor la opción más segura, pero no significa que sea la correcta.

Debido a esta situación, barajamos la existencia de otra variable necesaria para la obtención de la probabilidad, la información que nos proporciona las casillas colindantes o evidencias de una casilla no descubierta. Esta información la usaremos como la suma de evidencias.

Teniendo en cuenta que puede darse una casilla no descubierta con 8 evidencias a su alrededor, el máximo de la suma de vecinos será el quíntuple de números de vecinos.

Reuniendo toda esta información, procedemos a crear nuestro grafo:

De esta forma obtendremos el grafo que dependerá tanto de las filas, columnas, como minas. Una vez realizado, tendremos que unir un nodo con otro, es decir, crear las aristas que conecten una casilla con otra, para obtener las aristas, creando dos aristas, una que dependerá del número de vecinos y otra que dependerá de la suma de los vecinos.

Nuestra red bayesiana quedará de la siguiente forma:

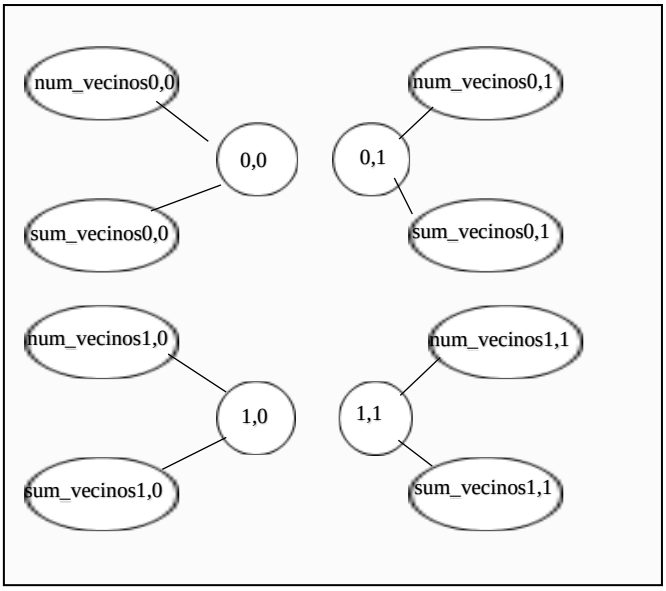


Ilustración 4. Red bayesiana para una matriz 2x2.

Con el fin de poder construir el modelo de la red bayesiana, tenemos que obtener primero cuáles serán los vértices que formen parte de una arista, para ello nos apoyaremos en el uso de la librería Pgmpy [7] que ofrece la clase BayesianModel, que proporcionando las aristas, los vértices se crearán automáticamente a partir de estos, teniendo en cuenta el uso del grafo para la obtención de los nodos.

Una vez que tenemos el modelo y éste contiene las aristas de este, haremos uso de un método, previamente definido, para definir la tabla de distribución de probabilidad condicional del modelo.

Para crear la distribución de probabilidad condicional, tendremos que calcular el número de vecinos que tendrá un nodo o casilla, teniendo en cuenta esto, aplicaremos el planteamiento anteriormente desarrollado en la que una casilla cuya suma de vecinos (la suma de la información que contiene una evidencia o casilla descubierta) sea mayor, tendrá más probabilidad de obtener una mina en la casilla sin desvelar, por lo que calcularemos dos tipos de probabilidad, con respecto a la suma de vecinos y con respecto a la suma de vecinos y número de vecinos, apoyándonos en ambas en la librería Numpy [6] que permite hacer cálculos entre arrays, por lo que obtendremos que:

$$probabilidad\ sum\_vecinos = \frac{1}{vecinos * 5 + 1}$$

$$probabilidad\ num\_vecinos = \frac{1}{vecinos + 1}$$

$$probabilidad\ por\ casilla_{sum\_vecinos,num\_vecinos} \begin{cases} t < vecinos + 1, \\ it = [0, vecinos * 5 + 1] \\ r = \frac{t}{vecinos} \\ res = \frac{\frac{r+it}{(vecinos*5),2}}{2} \end{cases}$$

siendo ***res*** *el conjunto de probabilidades de la suma de vecinos, por tantos vecinos que esta casilla tenga*

¿Pero cómo calculamos el valor de la información que nos proporciona una casilla descubierta?

El valor que se representa en una casilla o evidencia descubierta es la información que se nos proporciona informándonos de la cantidad de minas que existen en las casillas colindantes o vecinos. Estas variables son de tipo discreto y pueden tomar valores desde 0 hasta 8.

1) Casillas en las esquinas: Las casillas que se encuentren en las esquinas de los tableros, sólo pueden tomar los valores de 0 hasta 3, por lo que:

$$\begin{pmatrix} 0,0 & \cdots & 0,m-1 \\ \vdots & \ddots & \vdots \\ n-1,0 & \cdots & n-1,m-1 \end{pmatrix}$$

$$A: n = 0$$
$$B: m = 0$$
$$C: n - 1$$
$$D: m - 1$$

$$esquinas = (A,B) \cup (A,D) \cup (C,B) \cup (C,D)$$

2) Casillas en los laterales: Las casillas asociadas a los laterales en el tablero, sólo pueden tomar valores comprendidos desde 0 hasta 5.

$$laterales = A \vee B \vee C \vee D$$

3) El resto de casillas que no sean casillas en esquina o casillas en lateral, se considerará que el valor que pueden tomar será desde 0 hasta 8.

Teniendo en conocimiento todos los datos anteriores, procederemos al cálculo de la distribución de la probabilidad condicional para la probabilidad de obtener mina, creando una instancia de TabularCPD, una clase disponible en la librería Pgmpy [7], dónde proporcionaremos como datos el nombre de la variable, la cantidad de valores que puede tomar la variable, una lista de listas que contenga cada uno de las probabilidades para cada valor, según el valor de los padres, una lista con los nombres de los padres (num_vecinos y sum_vecinos), una lista con la cantidad de valores que puede tomar cada uno de los padres(num_vecinos: vecinos+1, sum_vecinos: vecinos*5_+1), hay que mencionar, que para obtener esta probabilidad hay que cuadrar los tamaños existentes entre los padres y la propia tabla. A su vez también crearemos otras dos tablas de distribución probabilística, dónde se almacenará siempre la misma información, en una la probabilidad de sum_vecinos y en otra la probabilidad de num_vecinos.

VI. Selecciona la casilla

Resuelto el cálculo de la probabilidad y sabiendo cuáles son las probabilidades dadas dependiendo del valor que tome una casilla, tenemos que proceder a aplicar el método de eliminación de variables en función del modelo de red bayesiana, el nodo o casilla sin descubrir y un diccionario que englobe el número de vecinos y la suma de vecinos que esta casilla tenga, para así saber exactamente que casilla tiene más o menos probabilidad de tener mina.

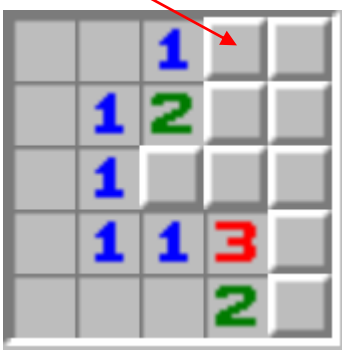
Pgmpy dispone de una clase llamada VariableElimination [8], ésta contiene un método llamado "query" que devolverá la probabilidad de contener mina o no con respecto al nodo o casilla sin descubrir que estamos consultando.

Para nuestra solución hemos querido que el método elimina variable generado, nos devuelva únicamente el valor de la probabilidad de no obtener una mina. Se ha de destacar también, que hemos considerado necesario, aplicar una pequeña variación, quitarle 0,05 a la probabilidad, para que el resultado no llegue nunca a 100%. Una probabilidad es una posibilidad de que se dé, pero no significa que sea fiable y sea la única opción o solución posible.

Prácticamente tenemos resuelto nuestro problema, sólo nos queda un último paso, obtener la casilla que tenemos que seleccionar porque tenga menor probabilidad de contener mina.

Para ello, necesitaremos nuestro modelo de red bayesiana y una lista con las evidencias descubiertas. Esta lista de evidencias contendrá todas las casillas no descubiertas con su respectivo nodo, el número de vecinos y la suma de la información que contengan esos vecinos, es decir:

$evidencias[0] = [(0,3), 2, 3]$



$$evidencias = [[(0,3), 2, 3], [(0,4), 0, 0], [(1,3), 2, 3], [(1,4), 0, 0], [(2,2), 6, 9], [(2,3), 3, 6], [(2,4), 1, 3], [(3,4), 2, 5], [(4,4), 2, 5]]$$

Ilustración 5. Cómo se forma la lista de evidencias y qué contiene exactamente una evidencia.

A partir de nuestra lista de evidencias, construiremos un diccionario que contendrá 2 claves, 'num_vecinos + nodo' y 'sum_vecinos + nodo' y 2 valores respectivamente que serán tanto el número indicado anteriormente de vecinos como la suma de estos. De esta forma, se combinará el método para obtener la probabilidad de no tener una mina mediante eliminación de variables y la lista de evidencias proporcionada, iterando sobre cada resultado de probabilidad con respecto a cada nodo, obteniendo como resultado la probabilidad asociada a este, dándonos a conocer cuál es la casilla con menor probabilidad de contener mina y, por consiguiente, la mejor opción de casilla a clicar, que será aquella casilla o nodo que contenga el menor valor de probabilidad.

Según la solución a nuestro problema y en el ejemplo expuesto en la Ilustración 5, la casilla sugerida a descubrir sería (0,4) con una probabilidad de 0,95 de no obtener mina:

Ilustración 6. Casilla descubierta una vez sugerida por nuestro problema. Se puede comprobar que la casilla despejada no contiene mina

Y así sucesivamente hasta resolver el juego y ganar, aunque también puede darse la probabilidad que no se gane y se seleccione una casilla con mina, pues la probabilidad no es una ciencia exacta.

VII. Juega al buscaminas

Como parte de la resolución de nuestro problema, hemos querido crear una interfaz gráfica para el usuario, así podrá usar nuestro método de sugerencia de movimiento de una forma cómoda y sencilla, para ello hemos utilizado la interfaz gráfica Tkinter [9] en Python.

Se ha creado una ventana principal, dónde se le pedirá al usuario que inserte el número de filas, columnas y minas que tendrá el tablero que quiere resolver.

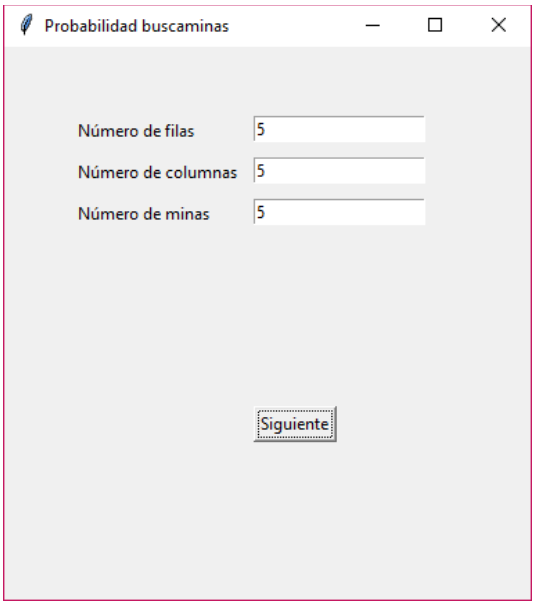


Ilustración 7. Ventana principal de la interfaz gráfica del problema.

Una vez que se introducen estos datos y se pulsa sobre el botón *"Siguiente"*, aparecerá otra ventana dónde se procederá a insertar en una lista las evidencias o información que se tiene en base a un juego del buscaminas:

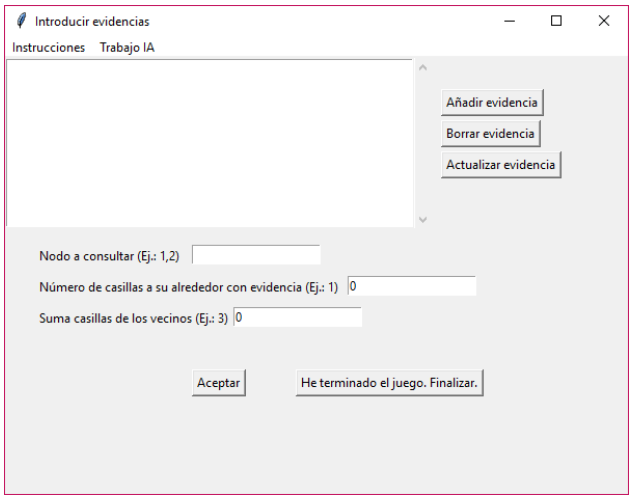


Ilustración 8. Ventana dónde se insertarán la información que tengamos acerca del juego.

El usuario dispone de instrucciones para saber cómo se ha de rellenar la lista de evidencias que le ayudará a calcular la probabilidad de las casillas sin descubrir, sugiriéndole una casilla como la mejor opción a clicar.

Consideramos que nuestra solución del problema es un apoyo al usuario a la hora de jugar, que le sirve como sugerencia probabilística, por lo que es necesario introducir siempre al menos una evidencia.

Como un extra a nuestra solución planteada, hemos querido usar una implementación ya realizada en Tkinter [3], para mayor comodidad al usuario a la hora de ejecutar el juego y nuestra implementación de sugerencia de casilla con menor probabilidad de tener mina. Esta implementación del buscaminas, se le han aplicado algunas modificaciones para adaptarlo un poco más al usuario, como traducciones en los mensajes.

VIII. Pruebas de rendimiento

Para llevar a cabo el experimento hemos realizado varias partidas con las distintas configuraciones propuestas en el informe hasta ganar la partida. En cualquier caso, se ha decidido establecer un número máximo de seis partidas por configuración.

Para medir el rendimiento del sistema hemos optado por comparar el número de aciertos obtenidos por cada iteración, entendiendo por iteración el número de veces que hemos clicado en el tablero del buscaminas, frente al número total de iteraciones. Esto nos aporta una proporción de 0 a 1 que nos permite medir el buen funcionamiento del sistema. En el apartado anexo adjuntamos una matriz de datos en la que hemos recogido esta información.

A continuación, se relacionan una serie de gráficos que reflejan la información contenida en dicha matriz de datos con respecto a las distintas configuraciones propuestas para medir el rendimiento del sistema. En estos gráficos el eje y representa el número de casillas despejadas, entendiendo por esto aquellas casillas que se desbloquean al clicar en el tablero del juego. El eje x representa el número de iteraciones (nº de clics). Además, se establece un código bicromático para indicar las iteraciones acertadas por el sistema y las falladas.

En la Ilustración 9 se observa una partida fallada en la tercera iteración y otra partida concluida positivamente.

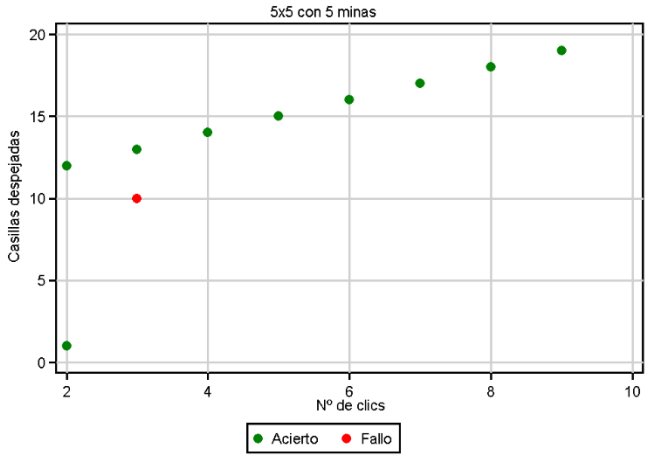


Ilustración 9. Gráfico de iteraciones para el tablero 5x5 con 5 minas

En la Ilustración 10, de nuevo se observa una partida fallada en la tercera iteración y otra partida concluida positivamente.

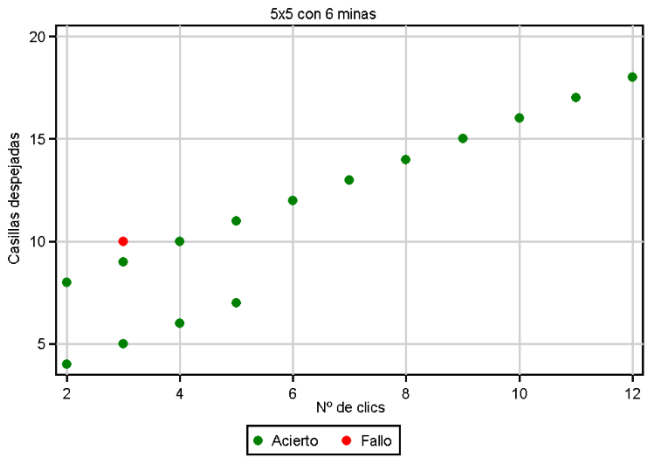


Ilustración 10. Gráfico de iteraciones para el tablero 5x5 con 6 minas.

A partir de la configuración 5x5 con 7 minas reflejada en la Ilustración 11, se observa en todas las configuraciones como hemos alcanzado el máximo establecido de 6 partidas falladas[1] precedidas del número de aciertos por configuración.

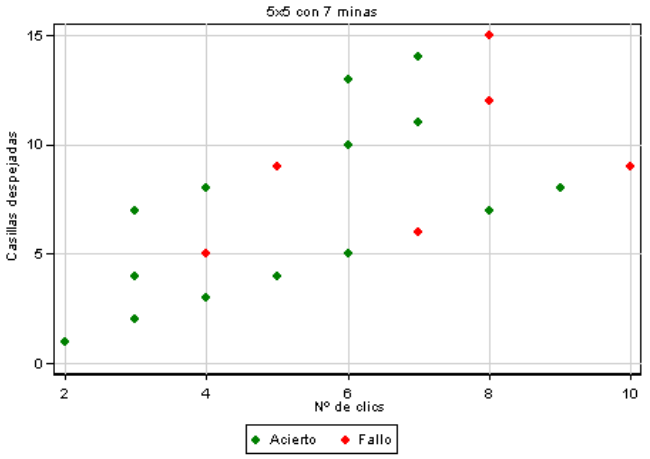


Ilustración 11. Gráfico de iteraciones para el tablero 5x5 con 7 minas.

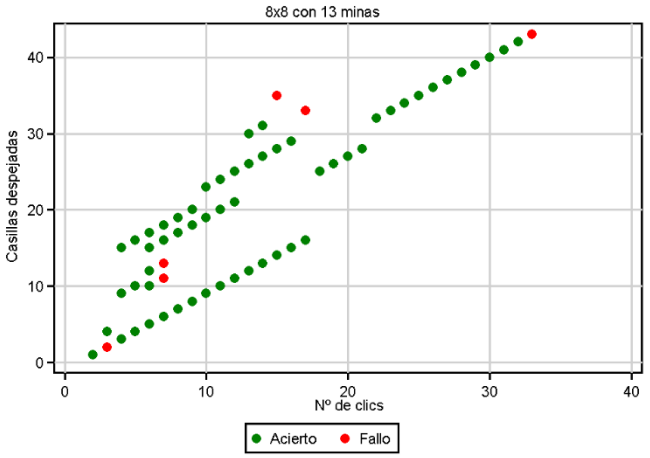


Ilustración 12. Gráfico de iteraciones para el tablero 8x8 con 13 minas.

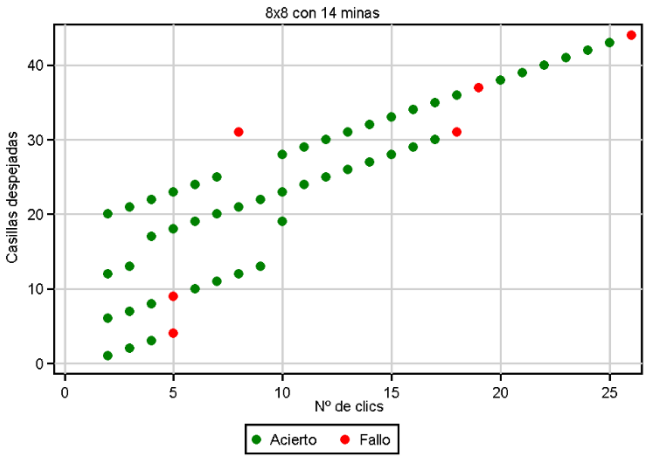


Ilustración 13. Gráfico de iteraciones para el tablero 8x8 con 14 minas.

[1] Nótese en la Ilustración 11, que en algunos casos hay iteraciones solapadas que afectan a su interpretación. Para conocer en detalle se remite a la matriz de datos anexa.

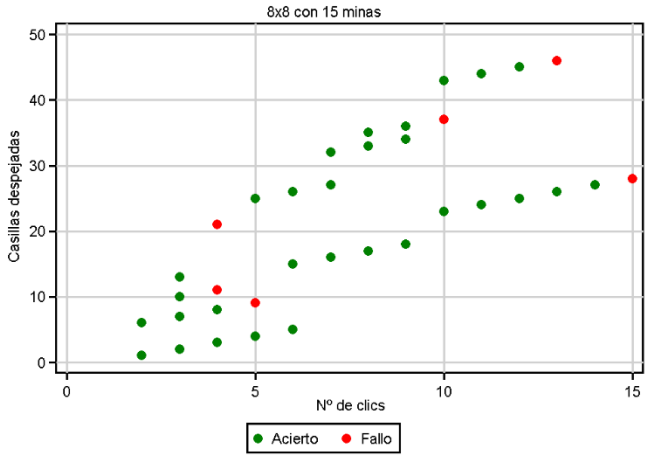


Ilustración 14. Gráfico de iteraciones para el tablero 8x8 con 15 minas.

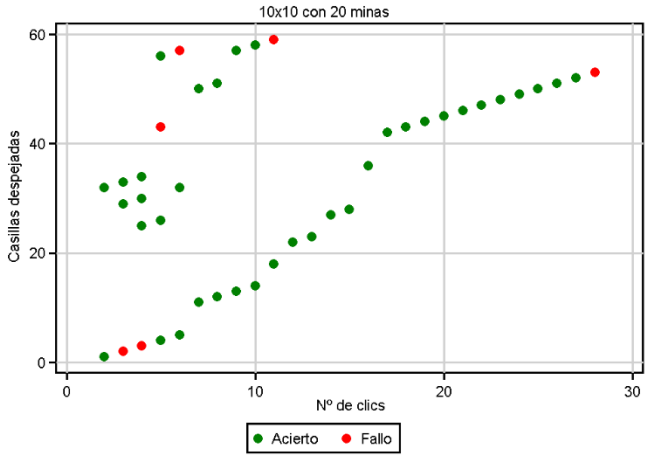


Ilustración 15. Gráfico de iteraciones para el tablero 10x10 con 20 minas.

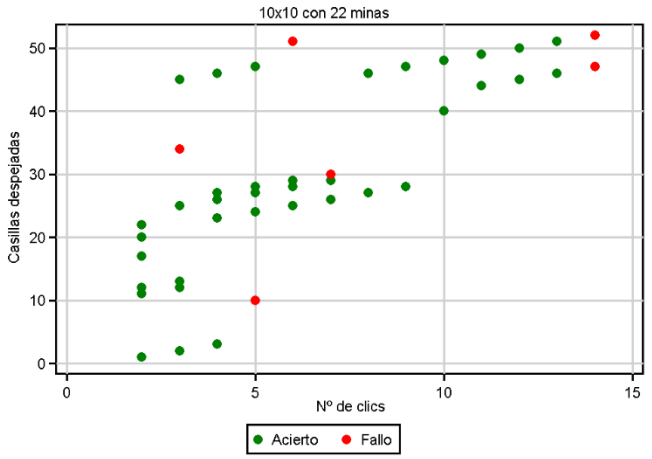


Ilustración 16. Gráfico de iteraciones para el tablero 10x10 con 22 minas.

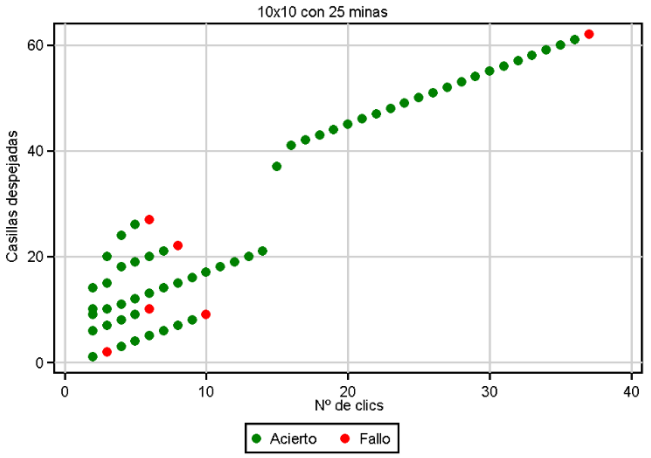


Ilustración 17. Gráfico de iteraciones para el tablero 10x10 con 25 minas.

En la Ilustración 18 se aprecia el rendimiento obtenido por el sistema para cada una de las configuraciones, teniendo en cuenta que este se ha establecido como el cociente del número de aciertos del sistema frente al total de iteraciones. En términos generales el rendimiento en todas las configuraciones oscila entre 0,83 y 0,92. Esto significa que todas las configuraciones tienden a tener un alto rendimiento de aciertos por casilla sin llegar a lograr ganar una partida.

En las configuraciones pertenecientes al tamaño de tablero 5x5, podemos observar cómo el rendimiento baja al introducir 7 minas. Lo mismo ocurre en las configuraciones con tamaño 8x8 con 15 minas. En el tamaño de tablero 10x10, sin embargo, se aprecia un rendimiento más bajo en la configuración con 22 minas.

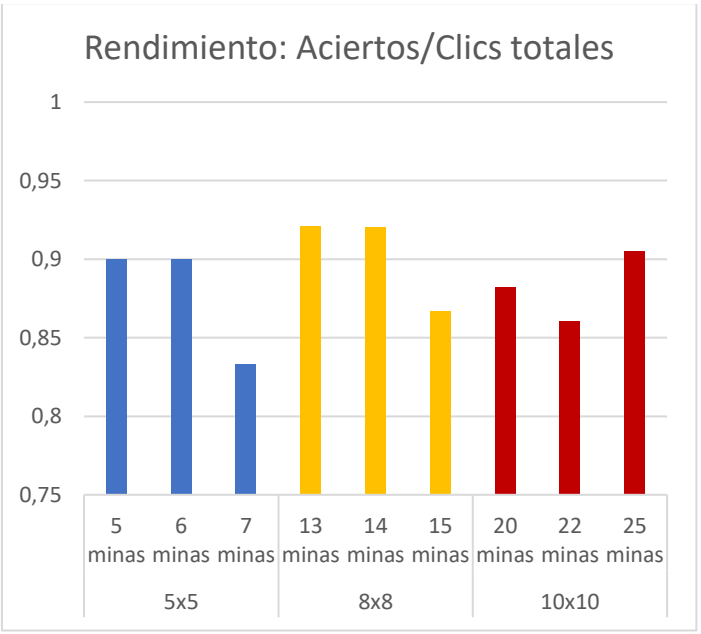


Ilustración 18. Gráfico resumen del rendimiento de todas las configuraciones.

IX. Conclusiones

El problema propuesto para dar solución al juego del buscaminas ha sido un ejemplo del posible uso real de las redes bayesianas para la resolución de problemas complejos.

Tras realizar las búsquedas de información necesaria para poder comenzar a divisar un posible planteamiento del problema, optamos por dibujar la red bayesiana que utilizaríamos. Este paso ha sido una labor muy costosa ya que no nos habíamos enfrentado con anterioridad a esbozar una red bayesiana. Poco a poco fuimos simplificando la red bayesiana pensando en cómo nos afectaría su tamaño, respecto a la idea de hacer un sistema de sugerencia de selección de la casilla más óptima para ganar la partida por parte de un usuario.

Con el esbozo de la red bayesiana comenzamos el desarrollo del sistema, en este ámbito la labor que más hemos tardado en realizar y comprender su funcionamiento ha sido calcular la distribución de probabilidad condicional.

Una vez realizadas las pruebas de rendimiento podemos concluir que la predicción de casillas con mayor probabilidad de no contener mina es buena, no obstante, el sistema no siempre llega a concluir satisfactoriamente una partida.

Debido a lo anteriormente explicado se considera como mejora al sistema volver a estudiar la red bayesiana para procurar una mejor distribución de probabilidad con el objetivo de progresar en la predicción de casillas sin minas con una mayor exactitud.

Asimismo, el desarrollo del sistema ha sido enriquecedor, por habernos podido enfrentar a tecnologías nuevas y desarrollos diferentes a los normalmente realizados.

Referencias

[1] J. L. Ruiz Reina, F. J. Martín Mateos et al. Redes bayesianas. Dpto. Ciencias de la computación e Inteligencia Artificial. Universidad de Sevilla. https://www.cs.us.es/cursos/iais-2017/temas/RedesBayesianas.pdf.

[2] M. Vomlelová, J. Vmiel. Applying Bayesian networks in the game of Minesweeper. Proceedings of the Twelfth Czech-Japan Seminar on Data aAnalysis and Decision Making under Uncertainty, pp. 153- 162 (2009) http://staff.utia.cas.cz/vomlel/vomlel-ova-cze-jap-2009.pdf.

[3] Minesweeper in Python Tkinter - https://codereview.stackexchange.com/questions/191550/minesweeper-in-python-tkinter.

[4] Minesweeper. https://www2.cs.duke.edu/courses/spring05/cps102/notes/minesweeper.pdf.

[5] MITCHELL, Michael N. A visual guide to Stata graphics. Stata Press, 2008.

[6] Numpy – Biblioteca de funciones matemáticas de alto nivel para operar con vectores o matrices en python. http://www.numpy.org.

[7] Pgmpy – Librería de python para trabajar con modelos gráficos y probabilidades. http://pgmpy.org.

[8] Pgmpy - VariableElimination. http://pgmpy.org/inference.html

[9] Tkinter – Biblioteca gráfica Tcl/Tk en python. https://docs.python.org/2/library/tkinter.html.