Bayes, profesional del Buscaminas

Daniel Díaz de Mayorga Ledesma
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
dandialed@alum.us.es
danieldml112@gmail.com

Walabonso Viejo Álvarez
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
walviealv@alum.us.es
walo19971997@gmail.com

El objetivo principal será el de aplicar algoritmos de inferencia exacta sobre un problema del videojuego *"Buscaminas"* modelado por redes bayesianas.

En cuanto a las conclusiones, se puede demostrar que usando este tipo de algoritmos aplicados al problema, podemos conseguir unos resultados óptimos, en cuanto a velocidad y ratio de victorias.

I. Introducción

El *Buscaminas* es un videojuego popularmente conocido por su aparición en las diferentes versiones del sistema operativo Windows. Este videojuego consiste en un tablero inicial de $n \times m$ casillas, las cuales el jugador debe ir destapando de forma que revele todas las casillas que no contengan ninguna mina de las que inicialmente se repartieron en el tablero.

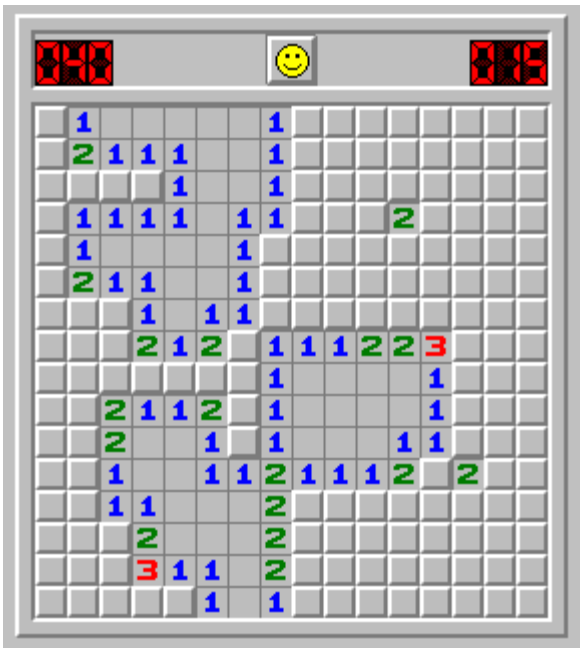


Fig. 1. Imagen del buscaminas de Windows

Debido a la relación directa que tiene el juego con la estadística y la probabilidad, siempre se ha tenido interés en la comunidad científica por conseguir algún tipo de sistema que resuelva estos tableros de forma automática con el mejor resultado posible. Este resultado dependería del número de derrotas por cada victoria que se consiga, y la velocidad con la que estas partidas se desarrollan.

Tras el estudio de las redes bayesianas en el campo de la Inteligencia Artificial, se ha propuesto conseguir el sistema mencionado con anterioridad a partir de este modelo. Junto con la aplicación del algoritmo de inferencia exacta de *eliminación de variables* se podrán**¡Error! No se encuentra el origen de la referencia.** obtener los resultados asociados al modelo.

La problemática asociada a este problema recae en el tamaño que puede adoptar la red bayesiana construida. Este modelo trabaja con múltiples nodos relacionados entre sí, de forma que al tener un número suficientemente alto de nodos se puede llegar a una complejidad tal que la eficiencia del sistema se vea negativamente afectada. Por esto, es interesante llevar a cabo una simplificación del modelo para optimizar los resultados.

II. Preliminares

A. *Métodos empleados*

Redes bayesianas [1]

Nuestra descripción del mundo vendría dada por un conjunto de variables aleatorias. Una variable aleatoria se podría definir como una "parte" del mundo cuyo estado podemos desconocer. Puede tomar diferentes valores de su dominio.

En nuestro problema se corresponden con:

- La variable aleatoria *Casilla$_{ij}$ tiene mina (Xij)* describe el hecho de que la casilla definida por la fila *i* y la columna *j* del tablero contenga una mina.
    - Valores posibles:
        - *Verdadero*: La casilla tiene mina.
        - *Falso*: La casilla no tiene mina.

- La variable aleatoria *Número de minas alrededor de la casilla$_{ij}$ (Yij)* describe, como bien denomina, el número que debería de aparecer en la casilla al ser destapa, que se corresponde con el número de minas que se encuentran en las casillas vecinas de la casilla definida por la fila *i* y la columna *j* del tablero.
    - Valores posibles:
        - *[0-8]*

Una red bayesiana es un grafo dirigido acíclico que consta de:

- Un conjunto de nodos, uno por cada variable aleatoria del "mundo".
- Un conjunto de aristas dirigidas que conectan los nodos; si hay una arista de X a Y decimos que X es un padre de Y (*padres*(*X*) denota el conjunto de variables aleatorias que son padres de X).
- Cada nodo Xi contiene la distribución de probabilidad condicional P (*Xi* | *padres* (*Xi*)). Esta distribución de probabilidad consiste en el grado de creencia sobre a, dado que todo lo que sabemos es que b ocurre, notada P(a | b).

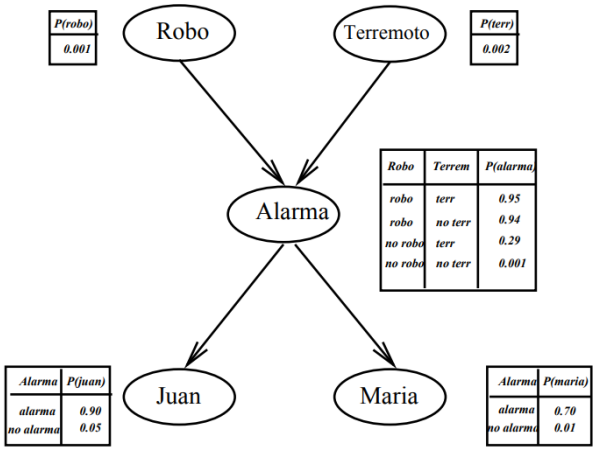


Fig. 2. Ejemplo de red bayesiana. [1]

Inferencia estadística [1]

Por inferencia estadística entendemos el cálculo de la probabilidad de una proposición dada condicionada por la observación de determinadas evidencias. Todas estas evidencias y distribuciones de probabilidad condicional se obtendrán del modelo de redes bayesianas.

Debido a las dimensiones que pueden llegar a alcanzar estos algoritmos sobre la red asociada, es necesario reducir los cálculos en la mayor medida posible. El método de Eliminación de Variables consigue reducir estos cálculos desechando aquellas variables que no tienen ninguna relación de dependencia con la variable a consultar, que junto con la agrupación y multiplicación de tablas (también llamadas factores) consigue minimizar aún más los cálculos necesarios.

B. *Trabajo Relacionado*

Existen varios artículos científicos relacionados con el acercamiento a la resolución de manera autónoma del juego del buscaminas.

Algunos de ellos se centran exhaustivamente tanto en el procedimiento de creación de la red bayesiana asociada al tablero, como en la inferencia estadística aplicada al modelo anterior [3].

Por otra parte, otros documentos están más orientados a la distribución de probabilidad condicional ligada a cada uno de los nodos de la red, junto con pequeños trucos demostrados experimentalmente a través de dichas distribuciones de probabilidad [4].

III. METODOLOGÍA

La finalidad del sistema a implementar es tal que, dado una instancia del juego del buscaminas, sea capaz de devolver la mejor casilla para destapar en el siguiente movimiento. Esta casilla deberá corresponder con la casilla que tenga menor probabilidad de contener una mina.

Para llevar a cabo este cometido, se procederán a usar toda la metodología indicada anteriormente.

Partimos de la red bayesiana formada por:

- Nodos:
    - *Xij*
    - *Yij*
- Aristas:
    - *padres*(*Yij*): {*conjunto_vecinos*(*Yij*)}

Conjunto_vecinos

Entrada:
- Nodo *Yij* (donde *ij* corresponde con la fila y columna en el tablero)

Salidas:
- Conjunto de nodos que se distancian de *Yij* en 1 fila y/o columna en el tablero

- Distribución de probabilidad condicional:

  - Para *Xij*:

$$P(x_{ij}) = \begin{cases} 1 & si \quad x_{ij} = Mina \\ \frac{totalMinas - minasEncontradas}{nCasillasNoReveladas} & e.o.c \end{cases}$$

$$P(\neg x_{ij}) = \begin{cases} 0 & si \quad x_{ij} = Mina \\ 1 - P(x_{ij}) & e.o.c \end{cases}$$

  - Para *Yij*:

Yij	$\sum Xij \ vecinos$	
$\forall i \in [0-8]$	$i == \sum xij$	1
	$e.o.c$	0

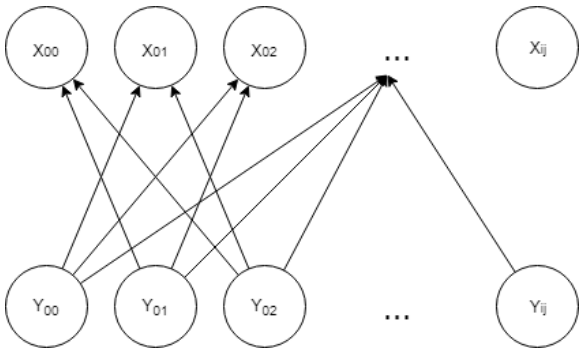


Fig. 3. Grafo de red bayesiana en buscaminas.

Con este tipo de red bayesiana, incluso en tableros de tamaño reducido, tendremos una cantidad de *i*×*j* nodos *Yij*, e *i*×*j* nodos *Xij*, relacionados siguiendo la función anterior. Esto da lugar a una complejidad lo suficientemente grande para que el algoritmo de inferencia utilizado sea incapaz de resolver una petición en un tiempo razonable.

Por ello se ha optado por simplificar la red de la siguiente forma:

- Partimos de *i*×*j* nodos totales, los cuales se dividirán en *Xij* o *Yij* dependiendo de su estado actual:
  - *Xij*: Se corresponderá con los nodos asociados a las casillas no descubiertas del tablero.
  - *Yij*: Se corresponderá con los nodos asociados a las casillas descubiertas del tablero.
- Al comenzar un tablero se destapará directamente la primera casilla que no sea mina. De esta forma evitamos crear una red donde solo haya nodos *Xij*, los cuales todos van a tener la misma distribución de probabilidad.
- Por cada nueva consulta tendremos que generar una nueva red bayesiana con los nodos *Xij* e *Yij* actualizados según su estado en el tablero resultante.

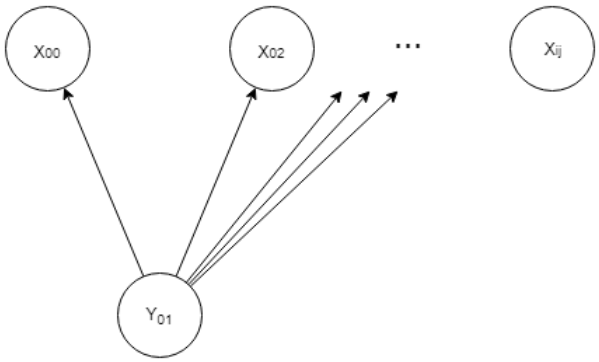


Fig. 4. Grafo de red bayesiana en buscaminas en la que solo se ha descubierto la casilla de la fila 0 y columna 1.

Con esta simplificación conseguimos reducir la red a la mitad de nodos, de forma que el algoritmo es capaz de conseguir resultados más eficazmente.

Además, de cara a la automatización del proceso para resolver un tablero completo, se pueden tener en cuenta las siguientes consideraciones:

- No preguntar por los nodos *Xij*, de los que ya sabemos de consultas a las redes anteriores del tablero, que la probabilidad de que contengan mina es 1.
- Si tras una consulta obtenemos varios nodos *Xij* que tienen una probabilidad nula de que contenga mina, destaparemos todas las casillas asociadas a cada uno de estos nodos.

Aunque se pueden considerar simplificaciones como dividir la red en pequeñas subredes, formadas solo por los nodos vecinos del nodo consultado, no se ha optado por ellas. Se debe tener en cuenta que dicha simplificación pierde veracidad en las probabilidades calculadas, lo que se traduce en una gran

reducción del tiempo de ejecución, pero el ratio de victorias se ve gravemente afectado.

Siguiendo la misma premisa anterior, se llegó a considerar otra simplificación en la que a la red anterior se le incluyeran los nodos vecinos a los ya vecinos del nodo consultado. Aun así, los resultados continuaban sin mejorar.

Teniendo en cuenta todo lo anterior, se ha procedido a la implementación del método que resuelva el problema.

Para dicha implementación se ha utilizado el lenguaje de programación Python [5], junto con la librería Pgmpy [6] para la generación del modelo de red bayesiano con sus tablas de probabilidad condicionales y el algoritmo de inferencia exacta por eliminación de variables.

Para facilitar el uso de la herramienta y generación de resultados, se ha implementado una versión personalizada del juego del buscaminas en lenguaje de programación Python.

Además, para facilitar la interacción con el usuario se ha implementado una interfaz gráfica en lenguaje de programación Java [7].

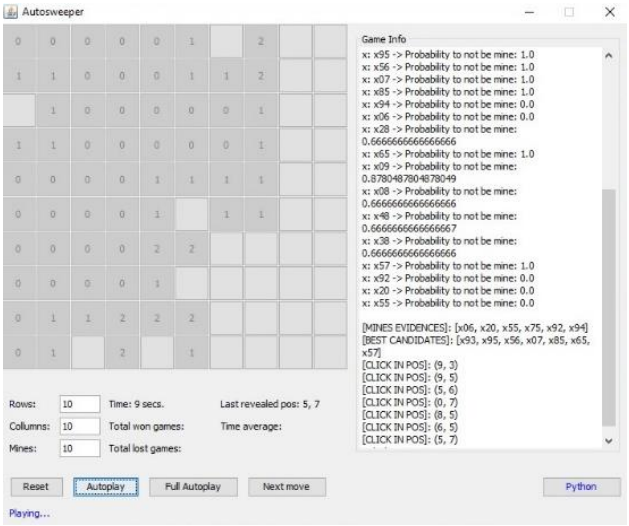


Fig. 5. Grafo de red bayesiana en buscaminas en la que solo se ha descubierto la casilla de la fila 0 y columna 1.

Tras obtener los resultados de los tests asociados al modelo propuesto, observamos que para tableros de hasta 8x8 los tiempos de ejecución eran razonables y los ratios de victoria óptimos. Sin embargo, para tableros de tamaño 10x10 o superiores el tiempo para resolver el tablero, e incluso para cada consulta, eran intolerables, del orden de más de 20 minutos por consulta.

Debido a esto, optamos por incluir ciertos cambios en la generación del modelo de red bayesiano para reducir los tiempos de consulta a valores lo suficientemente aceptables:

- Los nodos *Yij* los cuales no tengan vecinos de la forma *Xij* se ignorarán, debido a que toda su influencia en la red bayesiana ya ha sido completada, y no van a aportar ninguna evidencia relevante.
- El conjunto de nodos *Xij* los cuales no tengan vecinos de la frma *Yij* se agruparán en 1 sólo de ellos, no tienen ninguna relación (arista) con ninguno de los demás nodos de la red bayesiana, es decir, no son padres de ningún nodo, ni hijo de ningún otro.

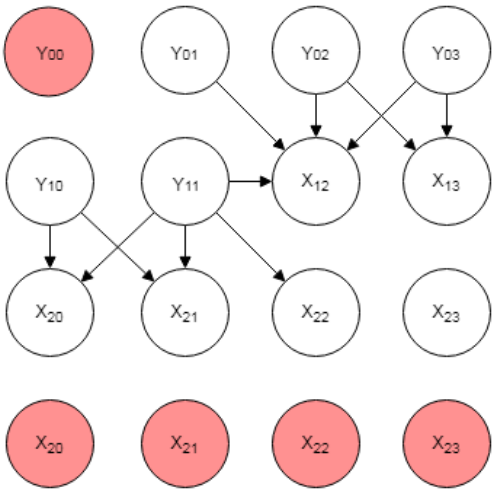


Fig. 6. Grafo de red bayesiana simplificada en buscaminas. Nodos en rojo ignorados para la generación de la red.

Tras esta simplificación, y los debidos tests, se pudo observar que el rendimiento mejoró de forma considerable, consiguiendo tiempos verdaderamente óptimos para todos los tamaños requeridos.

IV. Resultados

La forma de poder medir los resultados obtenidos por los métodos implementados se ha desarrollado con *logs*. Estos registros imprimirán por consola los datos relevantes del problema.

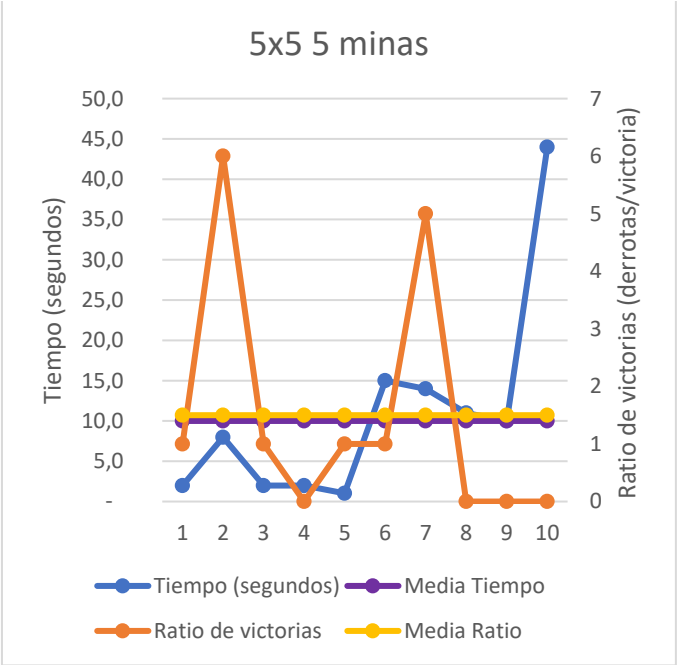
```
Board to resolve
0 1 ? ? ? ? ? ? ?
0 1 ? ? ? ? ? ? ?
0 2 ? ? ? ? ? ? ?
1 2 ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?


List of Xij nodes:
['x02', 'x03', 'x12', 'x22', 'x32', 'x40', 'x41', 'x42']

List of Yij nodes:
['y01', 'y11', 'y21', 'y30', 'y31']

List of Xij nodes that are known to be mine:
[]

Getting the best Xij with more probability to not be mine
x: x02 -> Probability to not be mine: 1.0
x: x03 -> Probability to not be mine: 0.7282608695652174
x: x12 -> Probability to not be mine: 0.0
x: x22 -> Probability to not be mine: 1.0
x: x32 -> Probability to not be mine: 0.0
x: x40 -> Probability to not be mine: 0.5
x: x41 -> Probability to not be mine: 0.5
x: x42 -> Probability to not be mine: 1.0

List of best Xij to not be mine: ['x02', 'x22', 'x42']
['x02', 'x22', 'x42']

Decided in 0.0 minutes
Revealing Positions: ['x02', 'x22', 'x42']
New board to resolve
0 1 2 ? ? ? ? ? ?
0 1 ? ? ? ? ? ? ?
0 2 4 ? ? ? ? ? ?
1 2 ? ? ? ? ? ? ?
? ? 4 ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
? ? ? ? ? ? ? ? ?
```

Fig. 7. Ejemplo de *logs* generado por un tablero de 10x10.

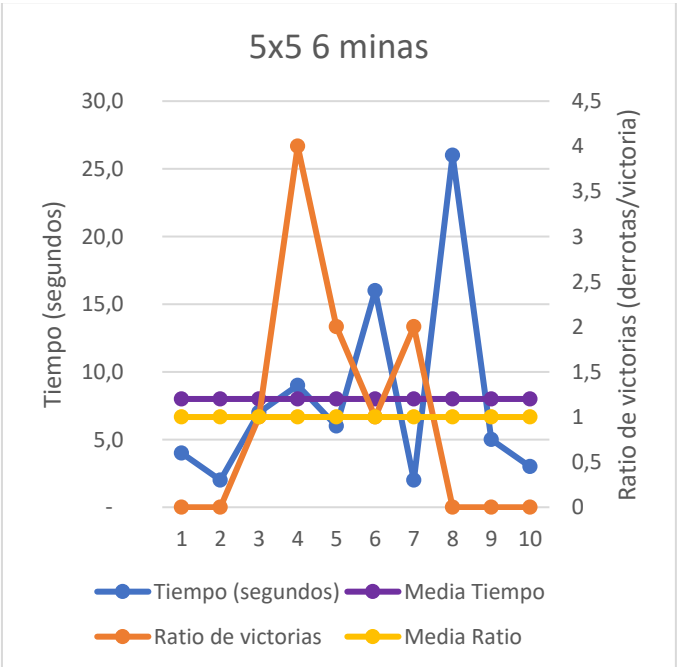
Para poder agilizar este sistema de *logs* y poder realizar los tests en segundo plano, se ha implementado un sistema de *logfiles* (archivos de registro). Se generará un archivo de texto en el que se escribirán todos los *logs* de forma que puedan consultarse en cualquier momento y poder realizar comparaciones con versiones futuras.

Los resultados obtenidos para los diferentes casos de prueba son los siguientes:

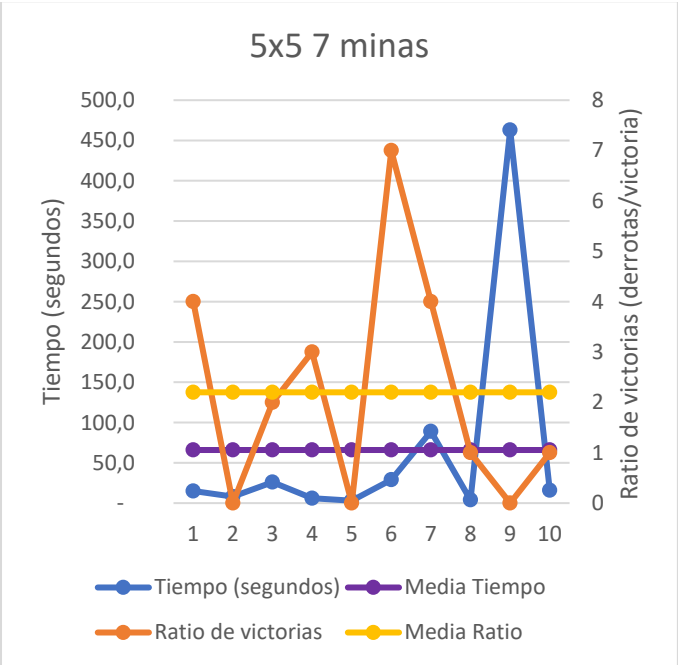
Gráfica 1. Tiempo de ejecución y ratio de victorias para 10 iteraciones de tableros de 5x5 con 5 minas



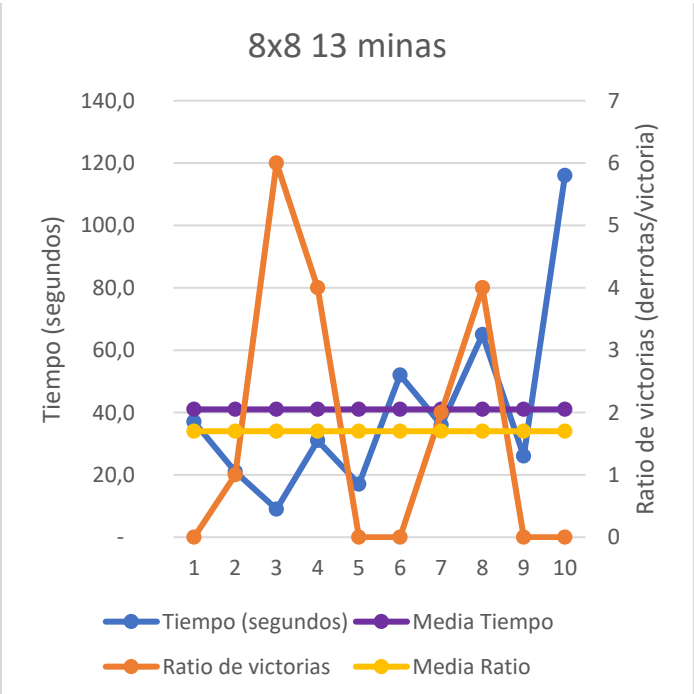
Gráfica 2. Tiempo de ejecución y ratio de victorias para 10 iteraciones de tableros de 5x5 con 6 minas



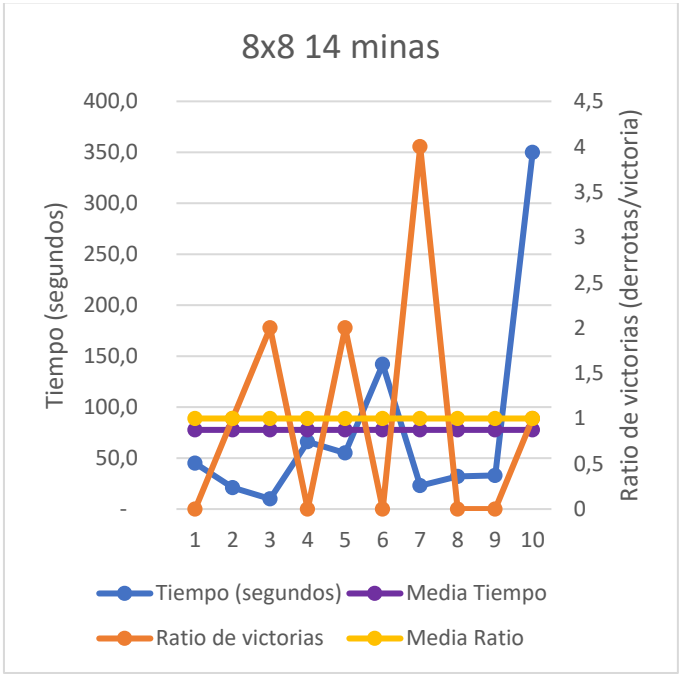
Gráfica 3. Tiempo de ejecución y ratio de victorias para 10 iteraciones de tableros de 5x5 con 7 minas



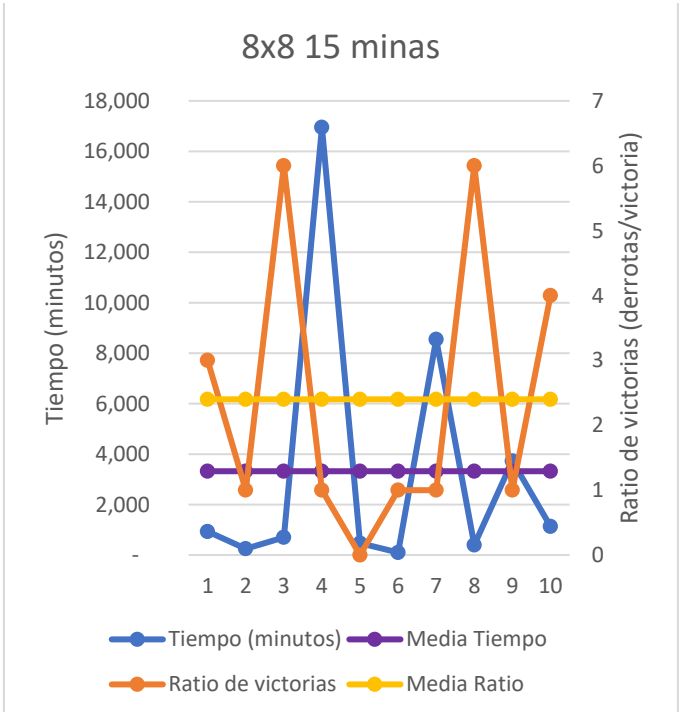
Gráfica 4. Tiempo de ejecución y ratio de victorias para 10 iteraciones de tableros de 8x8 con 13 minas



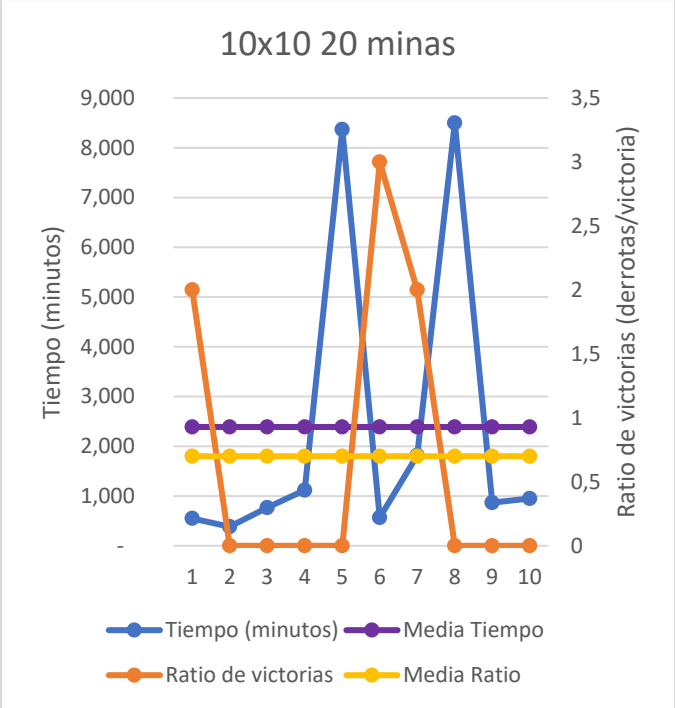
Gráfica 5. Tiempo de ejecución y ratio de victorias para 10 iteraciones de tableros de 8x8 con 14 minas



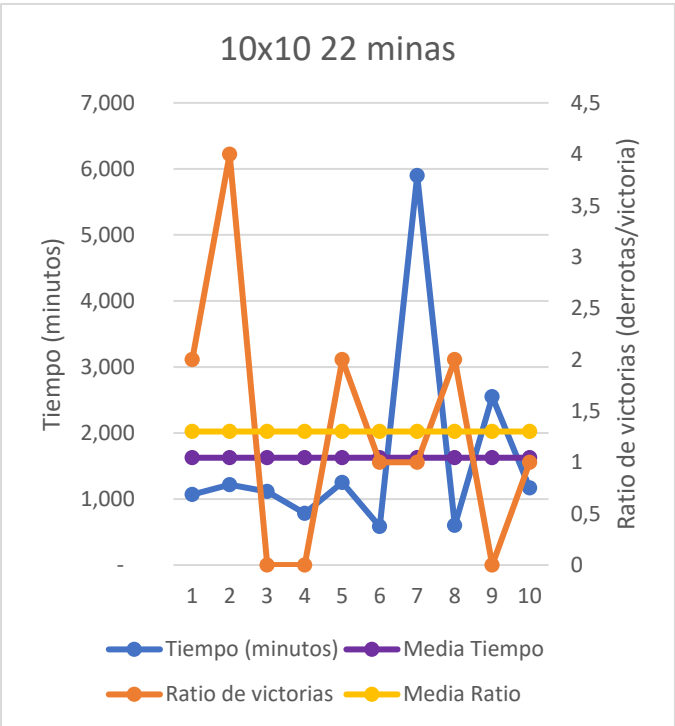
Gráfica 6. Tiempo de ejecución y ratio de victorias para 10 iteraciones de tableros de 8x8 con 15 minas



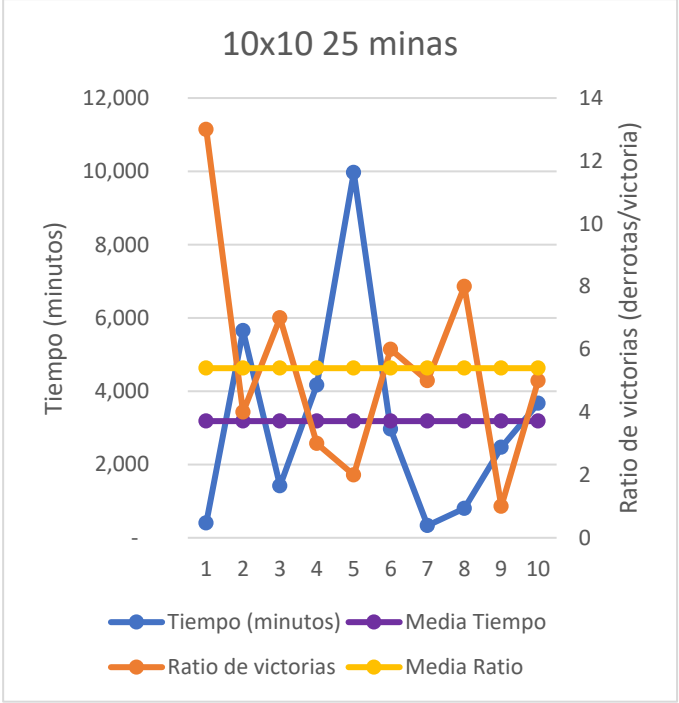
Gráfica 7. Tiempo de ejecución y ratio de victorias para 10 iteraciones de tableros de 10x10 con 20 minas



Gráfica 8. Tiempo de ejecución y ratio de victorias para 10 iteraciones de tableros de 10x10 con 22 minas



Gráfica 9. Tiempo de ejecución y ratio de victorias para 10 iteraciones de tableros de 10x10 con 25 minas



Con todos estos datos podemos observar que en los tableros de tamaño reducido el método funciona con una velocidad más que óptima y un ratio de victorias casi perfecto, teniendo algún desvío en algunas ocasiones debido a la aleatoriedad y dificultad de los tableros. Pueden llegar a darse situaciones en las que todas las casillas restantes por destapar tengan la misma probabilidad de contener o no una mina, por lo que la decisión será arbitraria.

Al acercarnos a los tableros con mayor tamaño observamos que los tiempos aumentan en un orden de pocos minutos, teniendo iteraciones en las que puede incluso no llegar al minuto de tiempo de ejecución. Al igual que en los casos anteriores, la aleatoriedad puede llegar a complicar el tablero, o en su defecto, simplificar la red de manera que los tiempos de ejecución sean mínimos.

En cuanto al ratio de victorias, sorprendentemente goza de una gran eficacia, con muestras de 0 derrotas por cada victoria, siendo perfecto, a excepción de situaciones como las indicadas anteriormente.

Todos los datos anteriores se corresponden a la última versión del método, con todas las simplificaciones presentadas en el apartado anterior. A continuación, se muestran algunas comparaciones de estos datos junto con los de los primeros métodos implementados.

Table 10. Comparaciones de resultados entre métodos implementados

Tablero		**Métodos implementados**		
		Método 1	***Método 2***	***Método Final***
5x5 5 minas	Tiempo	12 segs	10 segs	10 segs
	Ratio	3	2	1,5
8x8 13 minas	Tiempo	4,133 mins	3,283 mins	0,683 mins
	Ratio	7	5	1,7
10x10 20 minas	Tiempo	26,833 mins	7 mins	2.388 mins
	Ratio	27	13	0.7

Tras estas comparaciones podemos observar como las simplificaciones realizadas en el método han servido para mejorar los resultados y el grado de eficiencia y eficacia.

V. Conclusiones

Tras los resultados obtenidos por los tests, podemos llegar a decir que el método realizado por redes bayesianas e inferencia estadística exacta cumple con las expectativas de eficiencia al resolver un tablero del buscaminas. Un aficionado principiante puede llegar a resolver un tablero de 5x5 en unos pocos minutos, mientras que esta inteligencia artificial puede hacerlo en incluso menos de 5 segundos.

El estudio de las redes bayesianas para este tipo de problemas parece tener un sinfín de posibilidades, con respecto a la generación del modelo que proporcione la información para la posterior inferencia estadística. Ha sido interesante involucrarse en un problema de este estilo, intentando siempre poder mejorar los resultados implementando mínimas optimizaciones sobre el modelo de red bayesiano.

En cuanto a mejoras para un futuro, la interfaz gráfica de usuario podría mejorarse considerablemente para aclarar los datos mostrados y orientarla a un juego real de buscaminas, ya que su finalidad actual ha sido la de servir como entorno de pruebas.

Por último, la librería utilizada para la inferencia probabilística y la creación del modelo de red bayesiano podría ser revisada, ya que *Pgmpy* carece de potencia para problemas relativamente sencillos.

Referencias

[1] Tema 3 – Redes Bayesianas – Departamento de Ciencias de la Computación e Inteligencia Artificial – US (2017/2018) https://www.cs.us.es/cursos/iais-2017/temas/RedesBayesianas.pdf.

[2] Wikipedia – Algoritmo de eliminación de variables. https://es.wikipedia.org/wiki/Algoritmo_de_eliminaci%C3%B3n_de_variables

[3] Applying Bayesian Networks in the game of Minesweeper. http://staff.utia.cas.cz/vomlel/vomlel-ova-cze-jap-2009.pdf

[4] Bayes theorem and minesweeper. http://www.flyingcoloursmaths.co.uk/bayes-theorem-and-minesweeper/

[5] Página principal del lenguaje de programación Python https://www.python.org/

[6] Github del proyecto de la librería Pgmpy https://github.com/pgmpy/pgmpy

[7] Página principal del lenguaje de programación Java https://www.java.com/es/