UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE CIENCIAS

CIENCIAS DE LA COMPUTACIÓN

INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

Título del Trabajo

Problema de Coloración de Mapas modelado como
Problema de Satisfacción de Restricciones

Autores

Lázaro Camasca, Edson Nicks
León Rios, Marco Naro
Inocente Valle, Patrichs
Víctor Ponce, Pinedo

Profesor

Lara Avila, Cesar

Lima - Peru
(2018)

Contents

1 Objetivos

1.1 Objetivos Generales

- Solucionar el problema de coloración de mapas computacionalmente utilizando el **lenguaje R**.
- Verificar que se cumple el teorema de los cuatro colores.
- Breve introducción a la inteligencia artificial.

1.2 Objetivos Especificos

- Recopilar mapas habiles con terminacion **shapefile** "SHP".
- Optener la matriz de adyacencia del mapa.
- Implementar el algoritmo de coloracion que satiface el PSR.
- Mostrar el mapa coloreado.

2 Introducción

2.1 Inteligencia Artificial

2.1.1 Conceptos Generales

La inteligencia artificial es un área de estudio bastante nueva. El propio término de INTELIGENCIA ARTIFICIAL fue implantado en una conferencia en el año de 1956. Esta área aborda la simulación de procesos de inteligencia humana por parte de máquinas, especialmente sistemas informáticos. Estos procesos incluyen el aprendizaje (la adquisición de información y reglas para el uso de la información), el razonamiento (usando las reglas para llegar a conclusiones aproximadas o definitivas) entre otras. Durante el transcurso del tiempo han habido muchos científicos que han tratado de proporcionar una definición formal de IA. Muchas de estas definiciones se pueden agrupar mediante 4 enfoques distintos:

1. Sistemas que piensan como humanos
2. Sistemas que actúan como humanos
3. Sistemas que piensan racionalmente
4. Sistemas que actúan racionalmente

Las bases de la Inteligencia artificial son: Filosofía, Economía, Matemáticas, Neurociencia, Psicología, Computación, Linguística.

La Prueba de Turing

Para poder comprobar si efectivamente un sistema informático es realmente inteligente Alan Turing sugirió una prueba basada en la incapacidad de diferenciar entre entidades inteligentes indiscutibles y seres humanos. Un evaluador humano empieza a formular preguntas a un ser humano y a un sistema informático y estos dos brindarán sus respectivas respuestas. Si el evaluador humano concluye que no puede diferenciar cual de las respuestas provienen del sistema informático entonces se podrá decir que el sistema informático evaluado es inteligente.

Áreas de Aplicación de IA

- Tratamiento de Lenguajes Naturales: capacidad de traducción, órdenes a un sistema operativo, conversación hombre-máquina, etc.
- Sistemas Expertos: sistemas que se les implementa experiencia para conseguir deducciones cercanas a la realidad.
- Robótica: navegación de robots móviles, control de brazos móviles, ensamblaje de piezas, etc.
- Problemas de Percepción: visión y habla, reconocimiento de voz, obtención de fallos por medio de la visión, diagnósticos médicos, etc.
- Aprendizaje: modelización de conductas para su implante en computadoras.

Búsqueda heurística

Primero definamos lo que es una heurística. Una heurística es un algoritmo diseñado para encontrar la solución más óptima posible de un problema dado, sin embargo puede que si bien una solución se arroja de manera rápida, no se puede demostrar que siempre será así, o bien si la heurística nos devuelve la solución correcta no es posible demostrar que ante otros parámetros siempre devuelva la solución correcta. Las heurísticas están orientadas a reducir la cantidad de búsqueda requerida para encontrar una solución.

Ahora, debido a que muchos problemas y las etapas a seguir para resolverlos se pueden modelar mediante grafos, una búsqueda heurística nos ayudará a encontrar el camino más corto entre un nodo dado y el nodo objetivo que vendría a representar para nosotros la solución del problema. Para ello se definen funciones heurísitcas que representan que tan óptimo es ir hacia un nodo dado y en base a dicha función es que se van tomando las decisiones de que camino recorrer para poder llegar al objetivo.

Algoritmo A*

Este algoritmo es un metodo de solución para hallar el camino de menor coste en un grafo desde un nodo en específo hasta otro. Para lograr este resultado el algoritmo se vale de una función de evaluación de optimalidad f(n) y de funciones g(n) y h(n) donde:

- f(n): coste más barato estimado de la solución a través de n.
- g(n): coste del camino desde el nodo inicio al nodo n.
- h(n): función heurística que determina el coste estimado del camino más barato desde n al objetivo. Lo que hará un algoritmo en un nodo x determinado será analizar mediante la función f los nodos adyacentes a x y el nodo y escogido será el que nos devuelve el menor coste, es decir, el menor f(y) de todos.

2.1.2 Búsqueda entre Adversarios

El problema de búsqueda entre adversarios, también conocido como juego. El jugar a juegos fue una de las primeras tareas emprendidas en IA. Desde 1950 hasta la actualidad hubo un progreso continuo en el nivel de juego, hasta el punto de que las máquinas han derrotado a campeones humanos en ajedrez. Los juegos son interesantes porque son demasiado difíciles de resolver. Por ejemplo, el ajedrez tiene un árbol de búsqueda de aproximadamente 10^5 nodos por lo tanto se requiere la capacidad de tomar decisiones óptimas. Los juegos en inteligencia artificial, son aquel entorno en donde agentes tendrán que considerar las acciones de otros agentes, los agentes se encuentran en conflicto y ambos jugadores tratan de maximizar su rendimiento. Un juego puede definirse por el estado inicial (como se establece en el tablero), las acciones legales en cada estado, un test terminal (que dice cuándo el juego está terminado), y una función de utilidad que se aplica a los estados terminales.

Clasificacion de juegos según IA:

- Juegos de suma cero.
- Minimax o de dos jugadores.
- Por turnos.
- Deterministas.
- De información perfecta, como por ejemplo el Ajedrez.

En juegos de suma cero de dos jugadores con información perfecta, el algoritmo minimax puede seleccionar movimientos óptimos usando una enumeración primero en profundidad del árbol de juegos. El algoritmo de búsqueda alfa-beta calcula el mismo movimiento óptimo que el minimax, pero consigue una eficiencia mucho mayor, eliminando subárboles que son probablemente irrelevantes

2.1.3 Problema de Satisfacción de Restricciones

El PSR modela los problemas como una colección homogénea finita de restricciones sobre variables, las que son resueltas por métodos de satisfacción de restricciones. Las etapas básicas para la resolución de un problema PSR son su modelización y su posterior resolución mediante la aplicación de técnicas PSR específicas, que incluyen procesos de búsqueda apoyados con métodos heurísticos y procesos inferenciales. El modelamiento del problema, que permite representar un problema mediante un conjunto finito de variables, un dominio de valores finito para cada variable y un conjunto de restricciones que acotan las combinaciones válidas de valores que las variables pueden tomar. En el modelamiento de un PSR, es fundamental la capacidad expresiva, a fin de poder captar todos los aspectos significativos del problema a modelar. Ejemplos típicos de problemas que se pueden modelar como PSR son:

- El problema de las N reinas
- La coloración de mapas
- Sudoku
- Problemas de criptoaritmética

2.1.4 Machine Learning

Es un tipo de inteligencia artificial (AI) que proporciona a las computadoras la capacidad de aprendizaje automatico, sin ser programadas explícitamente. El aprendizaje automático se centra en el desarrollo de programas informáticos que pueden cambiar cuando se exponen a nuevos datos.

Modelos Lineales

Perceptron

Es la red de neuronas artificiales más sencilla. Está compuesta únicamente por una capa de neuronas de entrada y otra capa de neuronas de salida. El Perceptrón es una red capaz de aprender. En su configuración inicial a los pesos de las conexiones se les da valores arbitrarios, por lo que ante la presencia de estímulos,la red genera respuestas arbitrarias, respuestas que no coinciden con las deseadas, y no es hasta que los pesos se han ajustado de tal modo que la respuesta que emite es la deseada, que se considera que la red ha conseguido aprender.

K-Vecinos más cercanos

Es un algoritmo clasificador supervisado basado en Reconocimiento de patrones en criterios de vecindad, y también se conoce como algoritmo de clasificación k-NN. Parte de la idea de que una nueva muestra será clasificada a la clase a la cual pertenezca la mayor cantidad de vecinos más cercanos del conjunto de entrenamiento más cercano a ésta.

SVM

Son las siglas de Support Vector Machine. Este algoritmo supervisado se utiliza generalmente para solucionar problemas de clasificación. La idea del algoritmo es ser capaces de encontrar, con los datos de entrenamiento, un hiperplano que maximiza la distancia a las diferentes clases, lo que es conocido como el "margen máximo". Una vez hallado este hiperplano podemos usarlo para clasificar nuevos puntos. SVM tiene múltiples aplicaciones, por ejemplo para reconocimiento de imágenes, clasificación de texto o aplicaciones en el área de la biotecnología.

Arboles de decisión

Es una forma gráfica y analítica de representar todos los eventos (sucesos) que pueden surgir a partir de una decisión asumida en cierto momento. Nos ayudan a tomar la decisión "más acertada", desde un punto de vista probabilístico, ante un abanico de posibles decisiones.

Redes Neuronales

Consisten de unidades de procesamiento que intercambian datos o información,se utilizan para reconocer patrones, incluyendo imágenes, manuscritos y secuencias de tiempo (por ejemplo: tendencias financieras).

Tienen capacidad de aprender y mejorar su funcionamiento. Una primera clasificación de los modelos de redes neuronales podría ser, atendiendo a su similitud con la realidad biológica:

- El modelo de tipo biológico. Este comprende las redes que tratan de simular los sistemas neuronales biológicos, así como las funciones auditivas o algunas funciones básicas de la visión.
- El modelo dirigido a aplicación. Este modelo no tiene por qué guardar similitud con los sistemas biológicos. Su arquitectura está fuertemente ligada a las necesidades de las aplicaciones para la que es diseñada.

2.1.5 Proceso de Markov

Un proceso de Markov es un tipo especial de proceso estocástico que describe la evolución de un sistema de variables aleatorias con la particularidad de que la valor de la probabilidad de un estado depende solamente del valor de la probabilidad del estado anterior.

Mas formalmente, para el caso discreto: Un proceso de Markov, usualmente llamado Cadena de Markov, está definido por: Un conjunto de estado $S = \{1, ..., n\}$ Un conjunto de posibles

transiciones, a saber, los pares (i, j) para los cuales $p_{ij} > 0$, y, los valores numéricos para los cuales p_{ij} son positivos.
El proceso de Markov definido por una colección de variables aleatorias X_0,
$X_1, ..., X_n, ...$, que toman valores de S y que satisfacen $P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, ..., X_0 = i_0) = p_{ij}$

Para los estados $i, j \in S$ y las posibles secuenvias $i_0, ..., i_{n+1}$ de esos estados.

El rango de aplicación del modelo de Markov es extenso. Uno de ellos es el aprendizaje por reforzamiento

Aprendizaje por reforzamiento

Área del Machine Learning que toma ideas del conductismo en las cuales se determina las acciones que debe escoger un programa para obtener el mayor estimulo positivo de parte del entorno.

La situación donde el "estimulo" del ambiente depende, en una cierta medida mas no en toda, de las acciones que el agente realiza en el estado anterior puede modelarse mediante un proceso estocástico de Markov debido a que la respuesta del ambiente depende, en cierta medida, de las acciones del agente.

Dentro de todos los temas mencionados anteriormente se escogio el PROBLEMA DE SATISFACCIÓN DE RESTRICCIONES

3 Cuerpo

3.1 Definición de un Problema de Satisfacción de Restriciones

Un promebla de satisfacción de restricciones(o PSR) está definido por:

- Un conjunto de **variables**, $\{X_1, X_2, ..., X_n\}$
- Un conjunto de **dominios**, $\{D_1, D_2, ..., D_n\}$
- Un conjunto de **restricciones**, $\{C_1, C_2, ..., C_n\}$

A cada variable X_i le corresponde un dominio no vacio D_i, los dominios son los valores posibles que puede tomar la variable.

Estado

Un estado se define como una asignación de valores a una o todas las variables, $\{X_i = v_i, X_j = v_j ...\}$. A un estado que no viole ninguna restricción se le llama estado **consistente**. Un estado **completo** es una asignacion en que se menciona cada variable, es decir que se asigne valores a todas las variables.

Solución a un PSR

Es un estado completo y consistente. Es decir, es la asignación de valores a todas las variables de tal forma que satisfagan todas las restricción.

3.2 Problema de Coloración de Mapas

Dentro del Problema de Satisfacción de Restricciones se escogio el problema de coloración de mapas.
El problema de coloración de mapas es un problema que se puede modelar como un PSR.

Definicion

El Problema de coloración de mapas es aquel problema donde se tiene un **conjunto de colores** y un **mapa plano** dividido en regiones. El objetivo es colorear cada región del mapa de manera que regiones adyacentes**(frontera)** tengan distintos colores.

3.3 Teorema de los 4 Colores

El teorema menciona que basta solo 4 colores para colorear cualquier **mapa geográfico plano**, de modo que dos regiones con **frontera** común tengan diferente color. Además existen condiciones como:

- El mapa debe ser **conexo** y cada una de las regiones también es conexa.
- Se considera frontera, cuando dos regiones se tocan en mas de un punto.

3.4 Resolución del PSR

La resolución de un Problema de Satisfacción de Restricciones consta de dos fases, que son:

3.4.1 Modelazión del Problema

La modelazión de un problema es una parte muy importante para la resolución de problemas, el modelado se realizara en términos de **variables**, **dominios** y **restricciones**.

Gracias al modelamiento del Problema de coloración de mapas como un problema de PSR, se podra demostrar el Teorema de los Colores con ayuda de un ordenador.

La forma de modelar el problema dentro del PSR es asociando lo siguiente:

- Una **variable** X_i por cada región del mapa.
- El **dominio** D_i sea el conjunto de colores disponibles.
- Una **restricción** C_i para cada par de regiones contiguas sobre la variable correspondiente que no permita la asignación de identicos valores a las variables.

Para una mejor ilustracion, supongamos que tenemos un mapa con 8 regiones $\{r_1, r_2, ..., r_8\}$, donde cada region r_i es adyacente a la region r_{i+1} donde $i = \{1, .., 8\}$, además las regiones deben ser coloreadas con los posibles colores Rojo, Azul, Negro. La modelacion PSR sería:

- Variables: $\{r_1 = v_1, r_2 = v_1, ..., r_8 = v_8\}$
- Dominio: $\{Rojo, Azul, Negro\}$, un color para cada variable.
- Restricciones: $\{r_i \neq r_{i+1}\}$

Para la demostración de el teorema de los cuatro colores se utilizara un **dominio** que contenga solo cuatro colores.

3.4.2 Procesar las restricciones

Una vez que hemos modelado el problema como un PSR, hay dos formas de procesar las restriccones:

- **Técnicas de consistencia**: basadas en la eliminación de valores inconsistentes de los dominios de las variables (es decir, que no verifican las restricciones impuestas).
- **Algoritmos de búsqueda**: se basan en la exploración sistemática del espacio de soluciones hasta encontrar una solución (o probar que no existe tal solución en caso contrario) que verifica todas las restricciones del problema.

Para la demostración del teorema de los cuatro colores se opto por utilizar Algoritmos de búsqueda, especificamente El Árbol de Búsqueda y Backtraking Cronológico.

El Árbol de Búsqueda

Las posibles combinaciones de la asignación de valores a las variables en un PSR genera un espacio de búsqueda al que se puede dotar de estructura para ser visto como un **árbol de búsqueda**. De esta forma, después podremos recorrerlo siguiendo la estrategia que queramos. La búsqueda mediante **backtracking**, que es la base sobre la que se soportan la mayoría de algoritmos para PSR, corresponde a la tradicional exploración en profundidad DFS en el árbol de búsqueda.

La forma más habitual de darle estructura de árbol pasa por asumir que el orden de las variables es estático y no cambia durante la búsqueda, y entonces un nodo en el nivel k del árbol de búsqueda representará un estado donde las variables $\{x_1, ..., x_k\}$ están asignadas a valores concretos de sus dominios mientras que el resto de variables, $\{x_{k+1}, ..., x_n\}$, no lo están. Podemos asignar cada nodo en el árbol de búsqueda con la tupla formada por las asignaciones llevadas a cabo hasta ese momento, donde la raíz del árbol de búsqueda representa la tupla vacía, donde ninguna variable tiene asignado valor alguno.

Los nodos en el primer nivel son $1-$tuplas que representan estados donde se les ha asignado un valor a la variable x_1. Los nodos en el segundo nivel son $2-$tuplas que representan estados donde se le asignan valores a las variables x_1 y x_2, y así sucesivamente. Un nodo del nivel k es hijo de un nodo del nivel $k-1$ si la tupla asociada al hijo es una extensión de la de su padre añadiendo una asignación para la variable x_k. Si n es el número de variables del problema, los nodos en el nivel n, que representan las hojas del árbol de búsqueda, son $n-$tuplas, que representan la asignación de valores para todas las variables del problema. De esta manera, si una $n-$tupla es consistente, entonces es solución del problema.

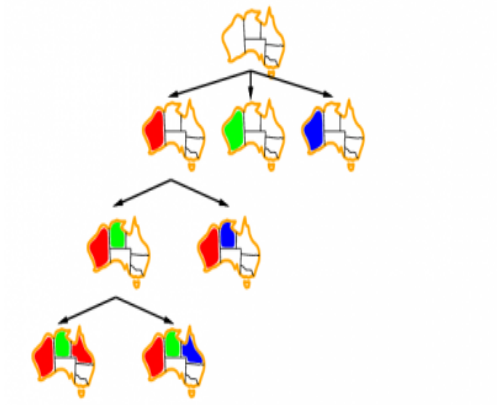

Figure 1: Árbol de búsqueda

Backtracking Cronológico

El algoritmo de búsqueda sistemática más
conocido para resolver PSR se denomina Algoritmo de Backtracking Cronológico (BT). Si asumimos un orden estático de las variables y de los valores en las variables, este algoritmo funciona de la siguiente manera:

Selecciona la siguiente variable de acuerdo al orden de las variables y le asigna su próximo valor. Esta asignación de la variable se comprueba en todas las restricciones en las que forma parte la variable actual y las anteriores: Si todas las restricciones se han satisfecho, vuelve al punto 1. Si alguna restricción no se satisface, entonces la asignación actual se deshace y se prueba con el próximo valor de la variable actual. Si no se encuentra ningún valor consistente entonces tenemos una situación sin salida (dead-end) y el algoritmo retrocede a la variable anteriormente asignada y prueba asignándole un nuevo valor. Si asumimos que estamos buscando una sola solución, BT finaliza cuando a todas las variables se les ha asignado un valor, en cuyo caso devuelve una solución, o cuando todas las combinaciones de variable-valor se han probado sin éxito, en cuyo caso no existe solución.

Es fácil generalizar BT a restricciones no binarias. Cuando se prueba un valor de la variable actual, el algoritmo comprobará todas las restricciones en las que sólo forman parte la variable actual y las anteriores. Si una restricción involucra a la variable actual y al menos una variable futura, entonces esta restricción no se comprobará hasta que se hayan asignado todas las variables futuras de la restricción.

BT es un algoritmo muy simple pero muy ineficiente. El problema es que tiene una visión local del problema. Sólo comprueba restricciones que están formadas por la variable actual y las pasadas, e ignora la relación entre la variable actual y las futuras. Además, este algoritmo es ingenuo en el sentido de que no recuerda las acciones previas, y como resultado, puede repetir la misma acción varias veces innecesariamente. Para ayudar a combatir este problema, se han desarrollado algunos algoritmos de búsqueda más robustos.

4 Estado del arte

5 Diseño del experimento

Describimos los objetos, funciones y tecnicas para solucionar el coloreo de mapas.

5.1 Objetos

Mapa: Contiene un archivo mapa en formato SHP.
MatrixAdyacencia: Contiene la matriz de adyacencia del mapa, considerando el mapa como un grafo.
Color: Es el vector que servirá para colorear las regiones del mapa, tienen una dimensión = n° de regiones.
El objeto Color solo puede tomar los siguientes colores:

- 0 = Blanco, es como si no le asignaramos el color.
- 1 = Negro
- 2 = Rojo
- 3 = Verde
- 4 = Azul

5.2 Funciones

Correcto: La función tiene como parámetros el objeto color y un entero. Aquí se comprueba si las variables cumplen con las restricciones, es decir si las regiones son adyacentes no deben tener el mismo color.

En el contexto del PSR el color representa las Variables, los colores el Dominio y la función Correcto las restricciones.

5.3 Técnicas

La técnica que se utiliza es la aplicación del **Algoritmo BackTraking**, ya que es uno de los algoritmos que más se ajusta para la resolución de problemas con Satisfacción de Restricciones. Para optimizar la búsqueda de la solución se le agregara **Heuristicas**.

La implementación del diseño del experimento se encuentra en el archivo Coloracion.R

Algorithm 1: Algoritmo de Backtracking Cronológico

Input: Vector de colores V
Output: Vector de colores V que satisface PSR

```
1  Procedimiento Backtracking(k, V[n]) # Llamada inicial: Backtracking(1, V[n])
2   Inicio
3   V [k] = Seleccion(d_k)        # Selecciona un valor de d_k para asignar a x_k
4   si Comprobar(k, V[n]) entonces
5       si k = n entonces
6           devolver V[n]         # Es una solución
7       sino
8           Backtraking(k + 1, V[n])
9       fin si
10  sino
11      si quedanvalores(dk) entonces
12          Backtraking(k, V[n])
13      sino
14          si k = 1 entonces
15              devolver ⊘        # Fallo
16          sino
17              Backtraking(k − 1, V[n])
18          fin si
19      fin si
20  fin si
21 fin Backtraking
```

6 Experimentos y resultados

Para los experimentos se utilizo dos mapas uno de Perú y Colombia de los archivos PaisPeru.shp y PaisColombia.shp

Los resultados obtenidos se encuentra en la **Figure 2** y **Figure 3**, se puede apreciar que la coloración de mapas cumple con las restricciones, además se utilizo un minimo de 4 colores.

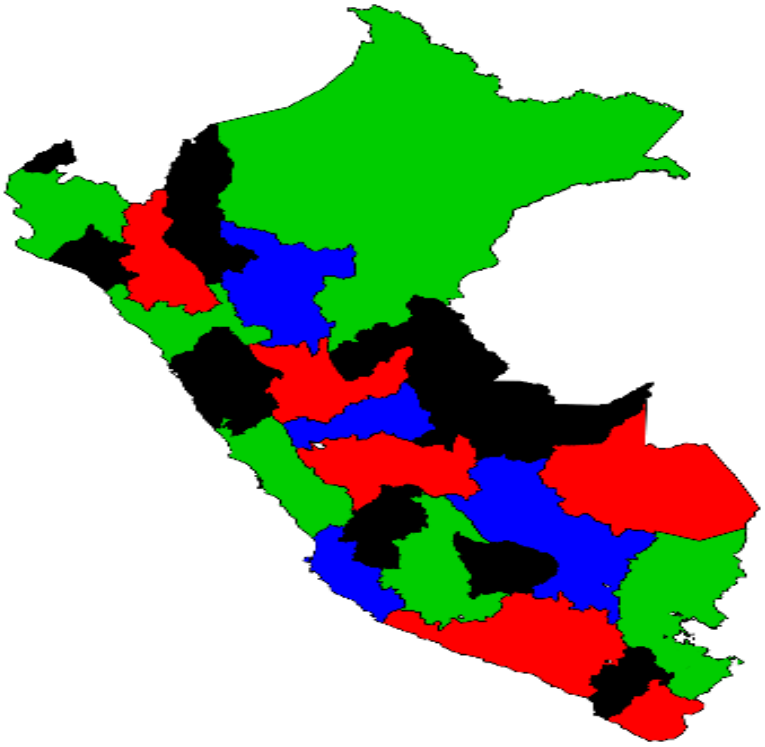

Figure 2: Coloración del mapa de Perú

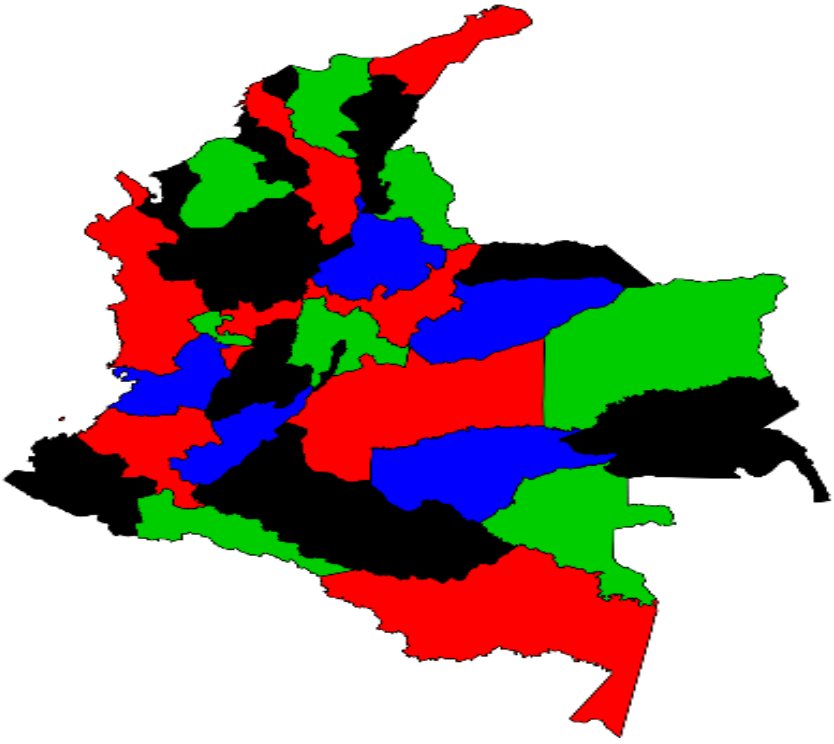

Figure 3: Coloración del mapa de Colombia

7 Discusión

7.1 Interpretación de los resultados optenidos

7.2 ¿Cómo se podria mejorar los resultados?

Para mejorar los resultados se puede implementar:

7.2.1 Algoritmos de Optimización

- **Algoritmos Look-Back**
  Los algoritmos look-back tratan de explotar la información del problema para comportarse más eficientemente en las situaciones sin salida. Al igual que el backtracking cronológico, los algoritmos look-back llevan a cabo la comprobación de la consistencia hacia atrás, es decir, entre la variable actual y las pasadas.
  Un algoritmo de este tipo es el **Backjumping (BJ)** este es parecido a BT excepto que se comporta de una manera más inteligente cuando encuentra situaciones sin salida.

- **Algoritmos look-ahead**
  Los algoritmos look-back tratan de reforzar el comportamiento de BT mediante un comportamiento más inteligente cuando se encuentran en situaciones sin salida. Sin embargo, todos ellos llevan a cabo la comprobación de la consistencia solamente hacia atrás, ignorando las futuras variables. Los algoritmos Look-ahead hacen una comprobación hacia adelante en cada etapa de la búsqueda, es decir, llevan a cabo las comprobaciones para obtener las inconsistencias de las variables futuras involucradas además de las variables actual y pasadas. De esa manera, las situaciones sin salida se pueden identificar antes y los valores inconsistentes se pueden descubrir y podar para las variables futuras.
  Uno de los algoritmos look-ahead más comunes es el **Forward checking (FC)**

7.2.2 Heurísticas

Los algoritmos de búsqueda para PSR vistos hasta el momento requieren el orden en el cual se van a estudiar las variables, así como el orden en el que se van a instanciar los valores de cada una de las variables. Seleccionar el orden correcto de las variables y de los valores puede mejorar notablemente la eficiencia de resolución. De igual forma, puede resultar importante una ordenación adecuada de las restricciones del problema.

Veamos algunas de las más importantes heurísticas de ordenación de variables y de ordenación de valores.

Ordenación de Variables

Muchos resultados experimentales han mostrado que el orden en el cual las variables son asignadas durante la búsqueda puede tener una impacto significativo en el tamaño del espacio de búsqueda explorado. Generalmente, las heurísticas de ordenación de variables tratan de seleccionar lo antes posible las variables que más restringen a las demás. La intuición indica que es mejor tratar de asignar lo antes posible las variables más restringidas y de esa manera identificar las situaciones sin salida lo antes posible y así reducir el número de vueltas atrás.

Las heurísticas de ordenación de variables estáticas generan un orden fijo de las variables antes de iniciar la búsqueda, basado en información global derivada del grafo de restricciones inicial.

Heurísticas de ordenación de variables estáticas

Se han propuesto varias heurísticas de ordenación de variables estáticas. Estas heurísticas se basan en la información global que se deriva de la topología del grafo de restricciones original que representa el PSR:

- **Minimum Width (MW)**
  La anchura de la variable x es el número de variables que están antes de x, de acuerdo a un orden dado, y que son adyacentes a x. La anchura de un orden es la máxima anchura de todas las variables bajo ese orden. La anchura de un grafo de restricciones es la anchura mínima de

todos los posibles ordenes. Después de calcular la anchura de un grafo de restricciones, las variables se ordenan desde la última hasta la primera en anchura decreciente. Esto significa que las variables que están al principio de la ordenación son las más restringidas y las variables que están al final de la ordenación son las menos restringidas. Asignando las variables más restringidas al principio, las situaciones sin salida se pueden identificar antes y además se reduce el número de vueltas atrás.

- **Maximun Degree (MD)**
  Ordena las variables en un orden decreciente de su grado en el grafo de restricciones. El grado de un nodo se define como el número de nodos que son adyacentes a él. Esta heurística también tiene como objetivo encontrar un orden de anchura mínima, aunque no lo garantiza.

- **Maximun Cardinality (MC)**
  Selecciona la primera variable arbitrariamente y después en cada paso, selecciona la variable que es adyacente al conjunto más grande de las variables ya seleccionadas.

8 Conclusión

- Se pudo dar una solucón al problema de coloracion de mapas utilizando la version 3.5.0 for Windows de R.

- Se pudo verificar el teorema de los 4 colores al aplicar nuestra solución a diversos mapas que se evaluaron en este informe.

- Se pudo recopilar mapas en formato **shapefile** "SHP", obtener su matriz de adyacencia con la función gIntersects() del paquete rgeos de R, implementar un algoritmo Backtracking con las modificaciones necesarias para satisfacer nuestro problema de satisfacción de restricciones y usar la funcion plot() para mostrar nuestros resultados de forma explícita.

9 Bibliografía

- Para la obtencion de mapas en formato con terminación **shapefile** "SHP" se utilizo,
  Paginas:
  http://www.datapages.com/gis-map-publishing-program/gis-open-files/global-framework/global-heat-flow-database/shapefiles-list

- Para el desarrollo del informe se utilizaron,
  Paginas:
  https://www.javatpoint.com/artificial-intelligence-in-robotics
  Libros:
  INTELIGENCIA ARTIFICAL UN ENFOQUE MODERNO, Russell, S.L, Segunda edición.