Búsqueda en espacios de estados

Carlos Alberto Valladarez Briones
Ingeniería de sistemas, Universidad de Cuenca
Cuenca, Ecuador
carlos.valladarez@ucuenca.edu.ec

***Abstract*— En el presente documento nos adentraremos en el funcionamiento de los métodos de búsqueda en espacios de estados, para esto analizaremos tanto métodos de búsqueda a ciegas, como heurísticos, esto se realizará con el fin de profundizar el conocimiento de estos algoritmos, y averiguar el mejor método basado en los problemas que han surgido a lo largo de su implementación y los resultados obtenidos.**

palabras clave: métodos, algoritmo, heurístico, búsqueda.

I. INTRODUCCIÓN

Mediante la teoría de grafos se analizarán diversos métodos de búsqueda en espacios de estados; en específico se analizarán los siguientes métodos de búsqueda a ciegas:

- Búsqueda en amplitud
- Búsqueda en profundidad
- Búsqueda bidireccional
- Búsqueda en profundidad iterativa
- Búsqueda de costo uniforme

En cuanto a los métodos de búsquedas a Heurísticas se analizarán los siguientes métodos:

- Método del gradiente
- Método primero el mejor
- Método A*

Esto se realizará con la finalidad de averiguar cuál método es el mejor en cuanto a complejidad espacial, temporal, y en cuanto a tiempo de ejecución.
Por otro lado también se indaga en dichos métodos con el fin de comprender estos algoritmos de búsqueda, y también para encontrar el mejor método posible, basándonos en los métodos obtenidos.

II. REPRESENTACIÓN DEL GRAFO

El grafo fue representado mediante un dataframe, el cual está estructurado mediante el siguiente formato:

Nodo origen	Nodo destino	Peso de la arista	Heurística del nodo origen

En el caso de que un nodo no esté conectado a otro la heurística del mismo se representará con el siguiente formato:

Nodo			Heurística del nodo

Esto se puede observar en la Fig 1. Además de esto se observa que en el caso del peso se llena con NA.

FROM	TO	weight	h
<fct>	<fct>	<int>	<int>
s	a	7	10
s	b	3	10
s	d	5	10
t		NA	4
g		NA	0

Fig. 1 Ejemplo de dataset cargado desde un archivo csv

Estos datos serán cargados de un archivo csv con el formato antes mencionado, aunque separado por comas, como se puede observar en la Fig 2.

```
FROM,TO,weight,h
s,a,7,10
s,b,3,10
s,d,5,10
t,,,4
g,,,0
```

Fig. 2 Ejemplo de archivo csv con el formato especificado

III.PROCESAMIENTO DEL GRAFO

Para la creación del grafo se utilizó la librería igraph, y la función graph_from_data_frame. Para lograr esto, primero se

guardó la heurística de los nodos en un vector. Posteriormente se eliminaron las filas con el formato:

Nodo			Heurística del nodo

Para definir el nodo final se creó un vector que contiene los nodos con la heurística igual a cero. Si se tratara del caso de una búsqueda bidireccional se tomará el primer valor del vector, en el resto de casos se realiza la búsqueda hasta que se encuentren todos los elementos del vector y que la pila o cola del método finalmente se encuentre vacío; pero se debe tomar en cuenta que sii el vector está vacío, se deberá recorrer el grafo con cada método hasta que la cola esté vacía, a excepción de la búsqueda bidireccional, en el caso de la búsqueda bidireccional se tomará un valor predefinido como final.

Una vez guardado el grafo se le adiciona el atributo heurística a los vértices del grafo mediante la función set_vertex_attr. Adicionalmente para la búsqueda bidireccional se creó un grafo similar al anterior, pero sin incluir la heurística, y haciéndolo no direccionado.

IV. EXTRACCIÓN DE DATOS DEL GRAFO

La librería igraph nos otorga una gran variedad de opciones para acceder a los datos almacenados en el grafo; las principales funciones utilizadas para extraer información fueron:

- Acceder a los hijos, o padres, de un nodo deseado: neighbors(grafo, nodo,modo)
- Obtener nombre del vértice:V(grafo)[index]$name
- Obtener el peso de una arista: E(grafo, P = (c(nodo actual,nodo destino)), path = NULL, directed = TRUE)$weight
- Obtener la heurística del vértice: V(grafo)[index/nombre]$heuristic

Además de esto se extrajo información para el cálculo de la complejidad computacional mediante bucles, o mediante funciones de igraph:

- Número máximo de sucesores
- Número de vértices
- Profundidad
- Número de aristas
- Promedio de sucesores

El grafo cargado, con el cual se realizarán las pruebas de los métodos de búsquedas se encuentra representado en la Fig 3.

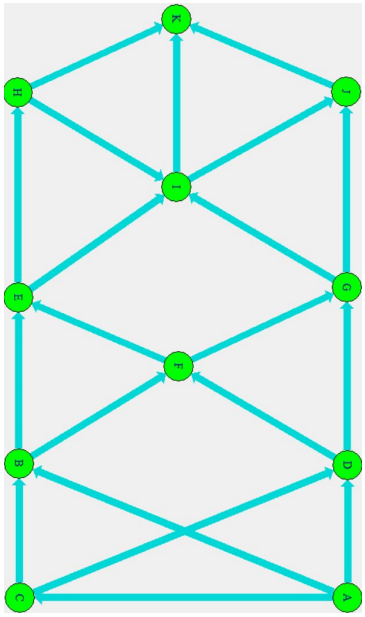


Fig. 3 Grafo representado mediante la función tkplot

V. BÚSQUEDAS A CIEGAS

- Búsqueda en amplitud

Para realizar este método de búsqueda se basó en el material compartido por el Ing. Victor Saquicela, por lo cual no se dio gran complicación; dicho método se basa en una cola en la cual se agregan todos los hijos del nodo actual, y de un vector que contendrá los todos nodos visitados; los nodos hijos que se encuentren ya en el vector de nodos visitados, no serán insertados en la cola.

Aparte de esto se agregó un condicional para que el algoritmo se detenga cuando se encuentren a todos los elementos del vector de búsquedas, o cuando la cola está vacía.

El recorrido de ejemplo se puede observar en la Fig 4.

```
[1] "Busqueda por Amplitud"
          Cola Extraer
1            A
2        C B D       A
3          B D       C
4        D F E       B
5        F E G       D
6          E G       F
7        G H I       E
8        H I J       G
9        I J K       H
10         J K       I
11           K       J
12 Encontrado->      K
```

Fig. 4 Resultado de la búsqueda por amplitud

- Búsqueda en profundidad

Este método resulta muy similar a de búsqueda en amplitud, sin embargo la única diferencia que presenta se encuentra en la utilización de una pila en lugar de una cola.

En la Fig 5. se puede ver el recorrido de este algoritmo en el grafo de ejemplo.

```
[1] "Busqueda en profundidad"
          Cola Extraer
1               A
2           D B C         A
3         G F B C         D
4       J I F B C         G
5       K I F B C         J
6         I F B C         K
7 Encontrado->            K
```

Fig. 5 Resultado de la búsqueda en profundidad

- Búsqueda bidireccional

La realización de este método se basa el uso de dos colas, la primera de estas recorre el grafo original desde el nodo de inicio, mientras que la segunda recorre un grafo no direccional que empieza desde el nodo final, en el caso de no existir un final, este será definido arbitrariamente. El algoritmo se detiene cuando uno algún valor de las dos colas coincide, en la Fig 6. se puede observar que coincide con el valor de "E"

```
[1] "Busqeuda Bidireccional"
[1] "COLA1"
        Cola Extraer
1           A
2       C B D         A
3     B D B D         C
4 D B D F E           B
[1] "COLA2"
          Cola Extraer
1             K
2         H I J         K
3       I J E I         H
4 J E I E G J           I
```

Fig. 6 Resultado de la búsqueda por amplitud

- Búsqueda en profundidad iterativa

Este método se basa en recorrer el grafo hasta el nivel de cada iteración, por ejemplo en la Fig 7. podemos observar que en la primera iteración recorre mediante el algoritmo de profundidad solo el nivel 1, es decir el nodo de inicio y sus hijos, en la siguiente iteración recorrerá hasta los hijos de estos últimos, y así sucesivamente hasta encontrar el nodo final o que no se encuentren nuevos hijos y la cola está vacía.

El principal reto de este algoritmo fue identificar el nivel actual del nodo; para lograr esto se creó un vector paralelo a la pila actual en la cual se coloca el nivel del nodo, de esta forma, si llegamos al nivel máximo de una iteración no se permite agregar hijos a la cola. una vez hecho esto solo es cuestión de aplicar el algoritmo de búsqueda en profundidad iterativamente hasta encontrar la respuesta.

```
[1] "Busqueda Iterativa en profundidad"
[1] "Nivel:"
[1] 1
   Cola Extraer
1     A
2 D B C         A
3   B C         D
4     C         B
5               C
[1] "Nivel:"
[1] 2
     Cola Extraer
1       A
2   D B C         A
3 G F B C         D
4   F B C         G
5     B C         F
6     E C         B
7       C         E
8               C
[1] "Nivel:"
[1] 3
        Cola Extraer
1           A
2       D B C         A
3     G F B C         D
4   J I F B C         G
5     I F B C         J
6       F B C         I
7       E B C         F
8         B C         E
9           C         B
10              C
[1] "Nivel:"
[1] 4
          Cola Extraer
1             A
2         D B C         A
3       G F B C         D
4     J I F B C         G
5     K I F B C         J
6       I F B C         K
7 Encontrado->          K
```

Fig. 7 Resultado de la búsqueda Bidireccional

- Búsqueda de costo uniforme

Este algoritmo se basa en encontrar el camino con menor peso

hasta encontrar la respuesta, para esto se utilizó un vector externo donde se guardo tanto el nombre como el peso hasta llegar a este nodo, este vector es ordenado mediante la función sort(), con el fin de analizar siempre el nodo que cuente con el menor peso total, esto se puede observar en la prueba representada en la Fig 8. , este proceso se realiza en cada iteración hasta que encuentra el resultado, o la cola queda vacía.

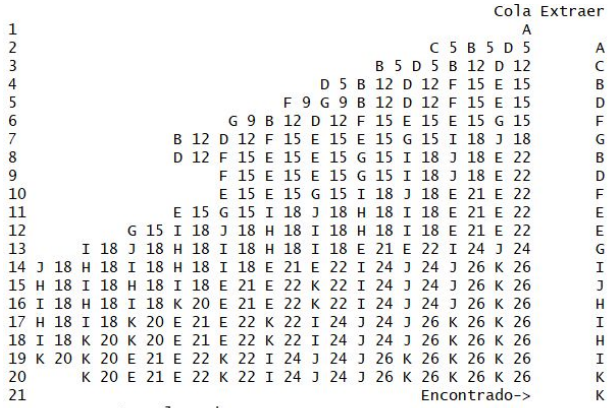


Fig. 8 Resultado de la búsqueda por costo uniforme

V.BÚSQUEDAS HEURÍSTICAS

- Método del gradiente

Este método se basa en tomar el nodo con menor valor que se encuentra en la cola, hecho esto se descarta la cola actual, y se analiza el nodo, se insertan los nodos hijos en un la y se repite el proceso hasta encontrar el nodo solución, esto se puede ver ejemplificado en la fig 9.

Para realizar esto se hizo algo similar a los dos últimos métodos analizados, se guardó el nombre y la heurística de los nodos hijos en un vector, el cual fue ordenado, una vez hecho esto se toma el menor y se vacía el vector.

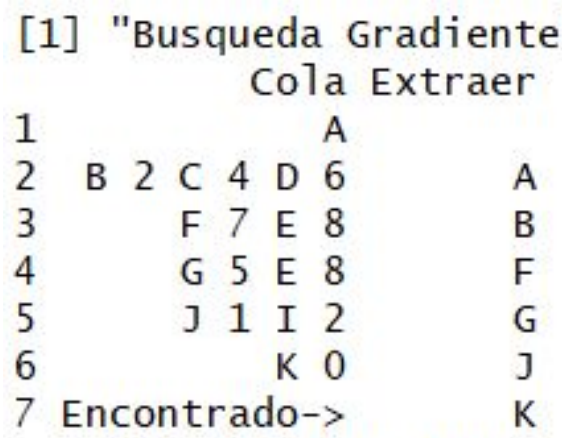


Fig. 9 Resultado de la búsqueda por el método del gradiente

- Método primero el mejor

Este método se basa en ir lo más profundo posible del grafo, hasta llegar a una respuesta o que no se encuentren más hijos, procurando tomar siempre el menor valor posible. Al no encontrar ningún sucesor del nodo actual, el algoritmo regresará al nodo padre de este nodo, y toma el siguiente hijo con mejor peso, entonces repite este proceso hasta encontrar una solución o hasta que la pila se vacíe.

Para lograr se analiza el nodo actual, se ordenan los hijos, y se agregan a la pila actual, como se puede observar en la Fig 10.

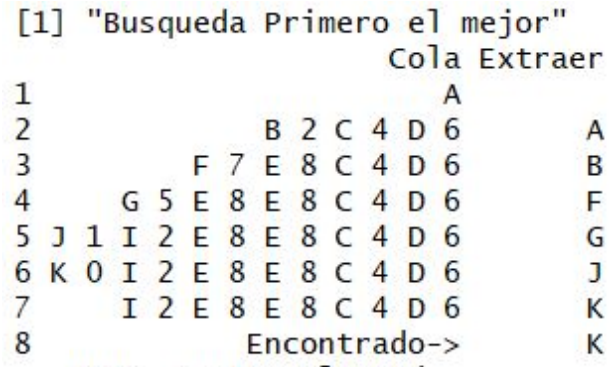


Fig. 10 Resultado de la búsqueda por el método primero el mejor

- Método A*

El método A* se basa en analizar el peso de la arista para llegar a un nodo hijo, junto con su heurística para encontrar el mejor camino.

Para esto se creó un vector en el cual se guardarán los tanto el nombre de los nodos como el valor del peso de la arista hasta el nodo objetivo, más el valor de la heurística, como se puede ver en la Fig 11. Si ya se visitó el nodo hijo no se lo agrega a la cola, si el nodo hijo se encuentra en la cola, no se lo agregara a menos que tenga un mejor valor.

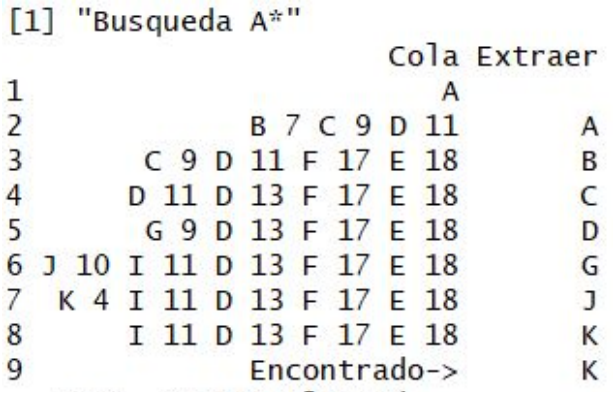
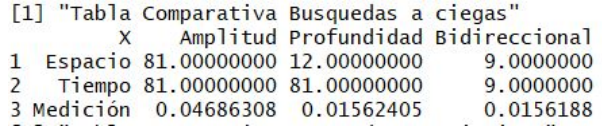


Fig. 11 Resultado de la búsqueda por amplitud

VI.ANÁLISIS DE TABLA COMPARATIVA

- Búsquedas a ciegas

En la Fig 12 podemos observar una tabla comparativa de la complejidad espacial, temporal, y una medición del tiempo de ejecución de cada método de búsqueda a ciegas.



Profundidad.Iterativa	Costo.Uniforme
12.0000000	81.0000000
81.0000000	81.0000000
0.1249702	0.1773689

Fig. 12 Tabla comparativa, búsquedas a ciegas

Podemos observar que en cuanto a complejidad espacial y temporal la mejor opción es la búsqueda bidireccional, en cuanto a tiempo de ejecución podemos observar que tanto la búsqueda a profundidad como la búsqueda bidireccional cuentan con los mejores tiempos. Por otro lado podemos observar que los peores métodos en cuanto a costo, tanto de espacio como temporal, son el método de búsqueda en amplitud, como el de costo uniforme. En cuanto a tiempo de ejecución los métodos con peores resultados son el de costo uniforme, como el de profundidad iterativa. En el caso de la búsqueda en profundidad una posibilidad de que el tiempo aumente es que se tiene que repetir el método de búsqueda en profundidad tantas veces como sean necesarias, lo cual ralentiza el proceso, una solución posible es guardar los hijos del nivel n en otro vector y agregarlas a la cola al terminar cada iteración, de esta forma en lugar de empezar la búsqueda desde cero se realizará directamente en sus hijos.

- Búsquedas heurísticas

En la Fig 13 podemos observar una tabla comparativa de la complejidad espacial, temporal,en el mejor y el peor caso, y una medición del tiempo de ejecución de cada método de búsqueda a Heurística

```
[1] "Tabla Comparativa Busquedas Heuristicas"
          X  Gradiente Primero.el.mejor  A.estrella
1  Tiempo MC 2.39789527       2.39789527 24.38652635
2  Tiempo PC 4.00000000      24.38652635 24.38652635
3 Espacio MC 1.00000000       1.00000000  1.00000000
4 Espacio PC 4.00000000      24.38652635 24.38652635
5   Medición 0.04686689       0.06248307  0.09378505
```

Fig. 13 Tabla comparativa, búsquedas Heurísticas

Al observar la tabla podemos apreciar que tanto el método del gradiente, como el de primero el mejor, cuentan con los mejores resultados en el mejor caso tanto de complejidad espacial, como temporal. En cuanto al peor caso podemos observar que la mejor opción en cuanto a tiempo y espacio es el método del gradiente, además de esto podemos observar que el método que cuenta con el mejor tiempo de ejecución resulta ser este método.

Por otro lado nos encontramos al método A*, que cuenta con los valores más altos en todos los apartados, siendo técnicamente el peor método.

Aunque estos datos pueden resultar engañosos, ya que en el método del gradiente, a pesar de aparentar ser la mejor opción en cuanto a complejidad, o tiempo, este puede verse atrapado en un bucle infinito , si no controlamos que no entre en nodos que ya han sido visitados. Además tenemos al método A* que a pesar de tener los peores resultados en la tabla este método asegura encontrar un resultado, es por esto que es muy utilizado.

VII. OBSERVACIONES ADICIONALES

En este apartado se resaltan algunas observaciones, o requerimientos que se encuentran fuera de los algoritmos de búsqueda.

- Menú: para realizar un menú en R se puede realizar mediante la función readine(), o se puede utilizar la función switch(), junto a la función menú(), como se puede observar en la Fig 14, para realizar un menú relativamente vistoso, y fácil de usar, y muy fácil de implementar, como se puede observar en la Fig 15

```
M=TRUE
while(M){
  switch(menu(c("CargarGrafo","Establecer nodo inici
         final(), CargarGrafo(),ReadStart(),AddEnd()
}
```

Fig. 14 Implementación de Menú

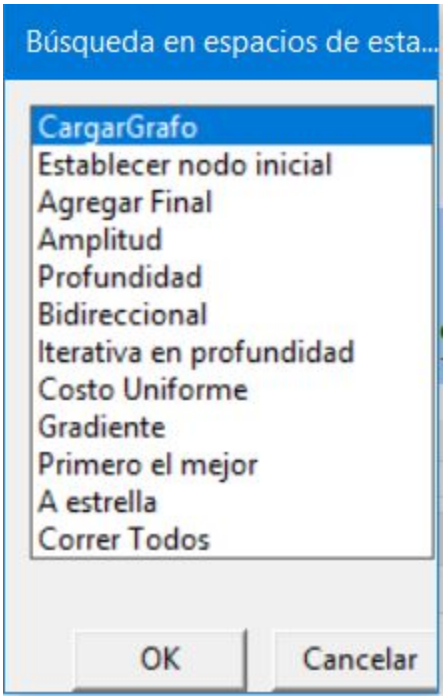


Fig. 15 Menú en ejecución

- Tiempos de ejecución: Para realizar esto de utilizar la función Sys.time() al inicio y al final de la ejecución de cada algoritmo.
- Tabla comparativa: Para realizar la tabla comparativa se crearon dos dataframes que contienen las ecuaciones necesarias para calcular la complejidad espacial, y temporal, así como los tiempos de ejecución de cada método.

VII.CONCLUSIONES

A lo largo de este documento hemos observado estos métodos de búsqueda, junto a su complejidad en cuanto a implementación, también a sus diferentes manifestaciones y reacciones de acuerdo al problema que se le presentó; comprobando así cada método. Se ha observado que los métodos de búsquedas a ciegas logran buenos resultados en grafos pequeños, además que estos resultan sencillos de implementar, aunque queda claro que al tratar con un grafo de

mayor tamaño estos métodos no son viables, dada la complejidad computacional que crece exponencialmente con la profundidad del grafo; dicho esto podemos percibir que los métodos heurísticos cuentan con una gran versatilidad, y poder, aunque no están exentos de problemas, ya que se pueden ver atrapados en bucles dentro del grafo, dando como resultado que no se encuentre un resultado especificado;

en el caso del método A* nos encontramos con un método potente, que nos asegura encontrar un resultado, con la desventaja de que, a comparación de otros métodos, este resulta más complicado de implementar, pero dados sus resultados en cuanto a complejidad computacional, tiempos de ejecución, y estabilidad, podemos decir que es el mejor método que hemos observado.

VIII. Bibliografía

[1]*Cran.r-project.org*, 2019. [Online]. Available: https://cran.r-project.org/web/packages/igraph/igraph.pdf. [Accessed: 18- Nov- 2019]

[2]"R como calculadora — OCW - UC3M", *Ocw.uc3m.es*, 2019. [Online]. Available: http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/r-como-calculadora. [Accessed: 18- Nov- 2019]

[3]"An Introduction to R", *Cran.r-project.org*, 2019. [Online]. Available: https://cran.r-project.org/doc/manuals/r-release/R-intro.html. [Accessed: 18- Nov- 2019]

[4]V. Saquicela, *Material proporcionado en clase*. 2019.