APLICACIÓN MÓVIL PARA EL CÁLCULO DE RUTAS "LOBOBICI" EN CIUDAD UNIVERSITARIA BUAP BASADA EN BÚSQUEDAS

Eliúh Cuecuecha Hernández
Benemérita Universidad Autónoma de Puebla
eliuhcueh@gmail.com

José Javier Martínez Orozco
Benemérita Universidad Autónoma de Puebla
javier.35.93.01@gmail.com

Daniel Méndez Lozada
Benemérita Universidad Autónoma de Puebla
daniel.mendezl@alumno.buap.mx

Adán Zambrano Saucedo
Benemérita Universidad Autónoma de Puebla
zambranos.adan@gmail.com

Aldrin Barreto Flores
Benemérita Universidad Autónoma de Puebla
aldrin.barreto@correo.buap.mx

Verónica Edith Bautista López
Benemérita Universidad Autónoma de Puebla
vbautista@cs.buap.mx

Salvador Eugenio Ayala Raggi
Benemérita Universidad Autónoma de Puebla
saraggi@ece.buap.mx

Resumen

En este documento se presenta la implementación de los algoritmos de búsqueda no informada: amplitud (BFS) y costo uniforme (UCS), mediante el

desarrollo de una aplicación móvil en la plataforma *MIT App Inventor*. La aplicación se enfoca en el cálculo del camino más corto y el de menos transbordos que el usuario puede realizar dependiendo de su estación de origen y destino. Con ello se obtuvo una herramienta para comparar el desempeño de ambos algoritmos a la vez que se le brinda al estudiante un mejor servicio en el programa LOBOBICI.

Palabras Claves: Aplicación móvil, app Inventor, búsquedas, búsqueda de costo uniforme (USC), búsqueda en amplitud (BFS).

Abstract

This paper presents the implementation of uninformed search algorithms: Breadth-first (BFS) and Uniform cost (UCS), by a mobile application development on the MIT App Inventor platform. The application focuses on the estimation of the shortest path and the less transfers the user can take depending on his origin and destination stations. This provides a tool to compare the performance of both algorithms while providing the student with a better service in the LOBOBICI program.

***Keywords*:** *App Inventor, breadth-first search (BFS), mobile application, searches, uniform cost search (UCS).*

1. Introducción

A lo largo de décadas se han estudiado problemas donde el objetivo es encontrar la ruta más corta [Guerriero, 2001]. Por muchos años diversos algoritmos de inteligencia artificial fueron propuestos para resolver este tipo de problemas que ahora se vuelve un reto determinar cuál es mejor para cada aplicación en específico [Kumar, 1988]. Las técnicas de búsqueda pretenden encontrar una solución válida dentro de un espacio de estados [Brooks, 1991].

En este documento se abordan las técnicas específicas de búsqueda en amplitud y costo uniforme, las cuales forman parte de las búsquedas no informadas. Son denominadas así debido a que el problema a resolver no ofrece ninguna

información adicional que ayude a encontrar una solución más rápido, es por esta razón que se han elegido estos algoritmos para el desarrollo de esta investigación.
Uno de los primeros artículos enfocado a resolver problemas del camino más corto es [Gallo, 1986]. En este documento se describen diferentes algoritmos de búsquedas desde un enfoque puramente matemático con el objetivo de dar una pequeña clasificación de los mismos sin ahondar en alguna aplicación en específico.
Se han elaborado múltiples trabajos comparativos de algoritmos de búsquedas informadas y no informadas. En [Chiong, 2008] se determina la ruta más corta en la isla de Borneo en Malasia con diferentes algoritmos de inteligencia artificial destacando la eficiencia del algoritmo Dijkstra. En este documento se resalta la importancia en estos días de encontrar la ruta más corta para llegar de un lugar a otro en poco tiempo debido a que la planeación de viajes por carretera es un asunto que cobra cada vez más relevancia gracias al exponencial aumento de vehículos. Se denota además cómo es que estos algoritmos son de cuantiosa aplicación en la resolución y soporte de muchos de los problemas cotidianos que tiene una sociedad.
En otros casos se optó por la comparación entre búsquedas no informadas contra heurísticas para la resolución de un rompecabezas y obtener una idea de cuál algoritmo tuvo mejor desempeño en base a la memoria consumida y el tiempo de ejecución [Kuruvilla, 2014]. En este mismo trabajo se presentan los parámetros que deben ser evaluados para poder discernir el algoritmo adecuado para cada aplicación y abre un panorama amplio de posibilidades para su implementación.
Hay trabajos que sólo abordan el plano teórico y nos brindan el *know-how* de cada algoritmo para enfocarlos en distintas aplicaciones, en ellos se denotan las características sobresalientes de estos métodos y dejan al usuario una primera aproximación de su desempeño como en [Chandel, 2014].
A pesar del vasto trabajo y desarrollo de estos métodos, no hay un documento registrado dónde se hayan puesto en marcha algoritmos de búsquedas en una aplicación móvil, debido principalmente a la baja capacidad de procesamiento que

tenían estos dispositivos. Ahora contamos con teléfonos celulares de iguales características que una computadora portátil convencional.

Cabe destacar que el buen desempeño de un método depende en mayor grado al tipo de aplicación para la cual se esté ejecutando, un algoritmo es eficiente sólo para ciertos casos y cumpliendo ciertos requisitos [Korf, 1999].

La aplicación desarrollada aborda el problema del desconocimiento de los caminos y rutas que la universidad tiene, cuando se es alumno de recién ingreso. El campus principal de la Benemérita Universidad Autónoma de Puebla es muy extenso y cuenta con pistas para circular en bicicleta a lo largo del mismo, este servicio se llama LOBOBICI. Muchas veces es difícil escoger el mejor camino para llegar de un punto a otro pues existen diversas posibilidades, el hecho de tomar una mala decisión puede llevar al estudiante a perder mucho tiempo.

La aplicación móvil ataca este problema directamente y compara la distancia a recorrer contra el número de estaciones por las que tiene que pasar el usuario. Lo anterior permite una manera interesante de contrastar el desempeño de los dos algoritmos previamente mencionados pudiendo destacar la eficiencia de la búsqueda por amplitud (BFS) frente a la de coste uniforme (UCS), para este caso en específico.

2. Métodos

En primer lugar, se generó un diagrama con las estaciones de LOBOBICI y la distancia entre cada una de ellas. Estas últimas se obtuvieron mediante el uso de la aplicación para *smartphone* SHealth de la empresa Samsung, que mediante el conteo de pasos estima la distancia recorrida con ayuda del GPS del dispositivo. El diagrama de la figura 1 muestra los datos recabados. Todas las distancias se expresan en metros y cada estación se encuentra numerada del 1 al 18.

Programación de la Aplicación para Android

El diseño e implementación de la aplicación, se realizó en MIT App Inventor. Esta es una herramienta de programación para la creación de aplicaciones en Android, la característica principal de ella es el uso de bloques que pueden

conectarse entre sí, simplificando el proceso de programación a una tarea más visual.

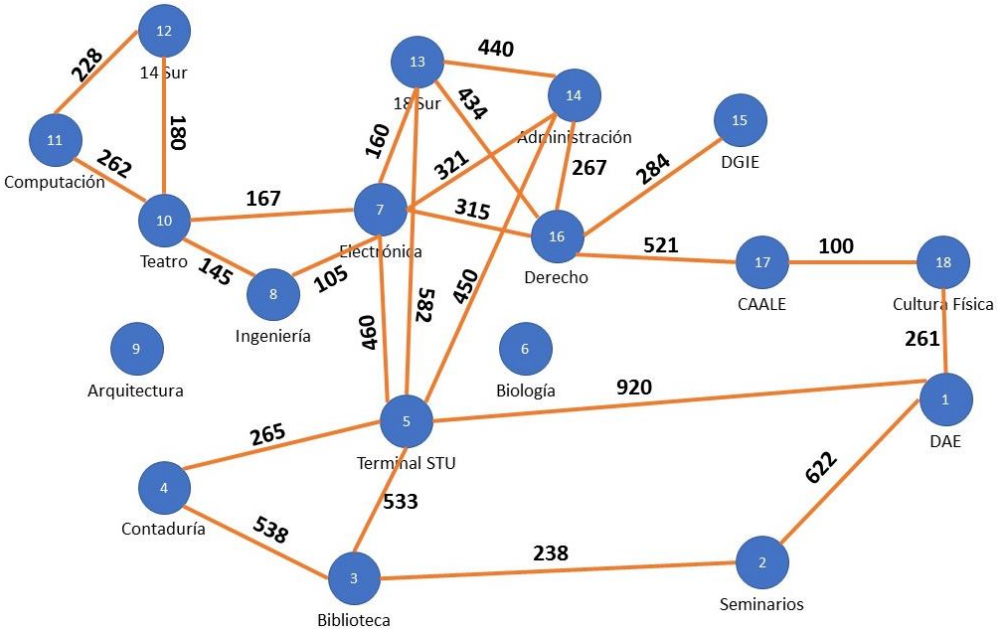

Figura 1 Esquema de las estaciones de LOBOBICI y distancia entre ellas.

Interfaz de Usuario

Esta consiste en una pantalla con el mapa de las estaciones de LOBOBICI de ciudad universitaria de la BUAP, se elige el origen y el destino para proceder a calcular la ruta más corta entre ellas, correspondiente a una búsqueda con costo uniforme (USC), o la ruta con menos estaciones que corresponde a una búsqueda por amplitud (BFS).

La figura 2 presenta el diagrama de bloques con el procedimiento para el uso de la aplicación.

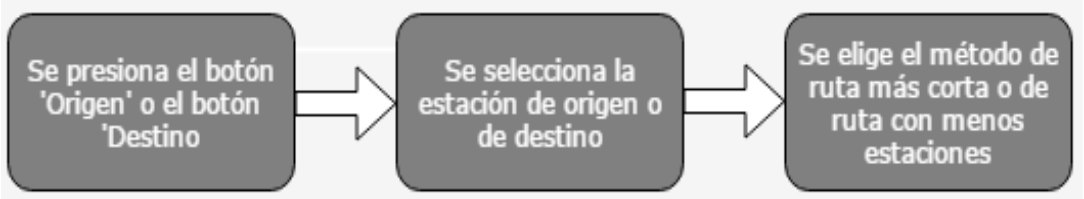

Figura 2 Diagrama de bloques de uso de la aplicación.

En la figura 3 se presenta la interfaz de usuario, los colores de las diferentes estaciones fueron establecidos por los administradores del programa LOBOBICI e indican las diferentes facultades que existen en el campus. Lo anterior sirve como guía visual para el usuario y no presenta relevancia para el desarrollo de este trabajo.

Figura 3 Interfaz de usuario.

Implementación de Algoritmo de Búsqueda por Amplitud (BFS)

El pseudocódigo mostrado en la figura 4, correspondiente al algoritmo de BFS [Serrano, 2012], hace uso de 2 listas principales, la de nodos frontera y la de nodos visitados. En la lista de nodos frontera se incluirán todos aquellos nodos que no han sido visitados ni repetidos. Mientras que la lista de nodos visitados contendrá a todos aquellos nodos que ya han sido visitados y que no son la solución [Korf, 1985].

Es importante mencionar que para el caso específico de las rutas de LOBOBICI, se descartan en primera instancia las estaciones de Arquitectura y de Biología pues ambas se encuentran cerradas debido a reparaciones que se llevan actualmente en el campus de la Universidad.

```
nodo_inicial = estado inicial
nodos_frontera = Cola FIFO
nodos_visitados = Lista
almacenar nodo_inicial en nodos_frontera
mientras nodos_frontera no vacío:
    nodo_actual = extraer un nodo de nodos_frontera
    si nodo_actual == solución:
        salir con solución

    introducir nodo_actual en nodos_visitados
    por cada operador:
        nodo_hijo = operador(nodo_actual)
        si nodo_hijo no en nodos_visitados ni nodos_frontera:
            introducir nodo_hijo en nodos_frontera
```

Figura 4 Pseudocódigo del algoritmo de búsqueda por amplitud–BFS [Serrano, 2012].

Además de las listas de nodos visitados y nodos frontera existe una lista que se encarga de almacenar los nodos padre con sus respectivos hijos. Esta lista sirve para obtener, a partir de una función programada en MIT App Inventor, el vector con los nodos que conforman el resultado.

Implementación de Algoritmo de Búsqueda por Costo Uniforme (UCS)

La organización de los datos en MIT App Inventor se realizó mediante listas, al igual que en el algoritmo BFS. En la figura 5 se presenta un fragmento de la organización de datos del mapa de la figura 1 en una estructura de tipo lista. La lista principal llamada datos_estaciones contiene sublistas que muestran las conexiones entre cada estación y la distancia que corresponde.

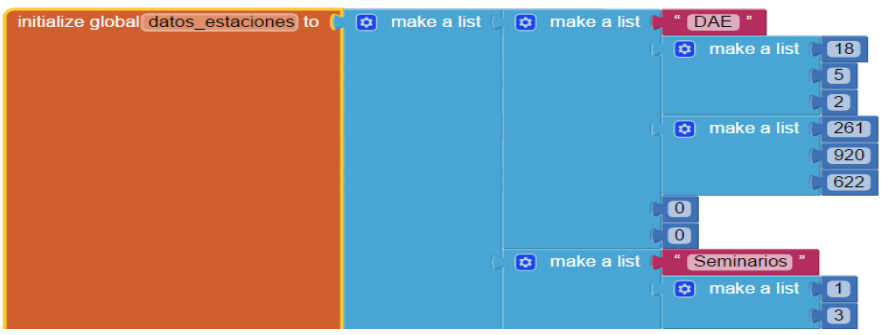

Figura 5 Datos del mapa en forma de lista en MIT App Inventor.

El pseudocódigo del algoritmo de UCS se muestra en la figura 6 [Serrano, 2012], en él se distingue el uso de dos listas, nodos_frontera y nodos_visitados. La primera corresponde a una cola con prioridad, cuyo uso marca la diferencia principal entre los métodos mostrados en el presente trabajo [Betali, 1999].

```
nodo_inicial = estado inicial
nodos_frontera = Cola con prioridad
nodos_visitados = Lista
almacenar nodo_inicial en nodos_frontera
mientras nodos_frontera no vacío:
      ordenar la lista de nodos_frontera según el costo
      nodo_actual = extraer el primer nodo de nodos_frontera
      si nodo_actual == solución

      introducir nodo_actual en nodos_visitados
      por cada operador:
            nodo_hijo = operador(nodo_actual)
            si nodo_hijo no en nodos_visitados:
                si nodo_hijo en nodos_frontera:
                    si coste de nodo_hijo < nodo en nodos_frontera:
                        sustituir nodo_hijo en nodos_frontera
                    si no
                        introducir nodo_hijo en nodos_frontera
```

Figura 6 Pseudocódigo de búsqueda por costo uniforme–UCS [Serrano, 2012].

3. Resultados

Una vez finalizada la implementación en MIT App Inventor, se procedió a verificar que los resultados obtenidos fueran los esperados; las pruebas se realizaron tomando en cuenta el mapa mostrado en la figura 1.

En las tablas 1 y 2 se muestran principalmente casos críticos, es decir, los casos cuyas estaciones de origen y destino se encuentran separadas de extremo a extremo en el campus. Estas pruebas también sirven para dar ejemplo de las diferencias que presentan ambos algoritmos.

Se evalúa también el tiempo de procesamiento de cada algoritmo para verificar cual llega a una solución más rápido, consumiendo menos recursos.

Las pruebas 1 a 7 describen casos críticos; en las últimas 3 se muestran ejemplos donde ambos métodos arrojan los mismos resultados, pero se puede observar una diferencia en el tiempo de procesamiento. En general cada prueba presenta una diferencia de 200 ms entre la búsqueda por amplitud y la búsqueda de costo uniforme, lo podemos notar en la tabla 2.

Tabla 1 Resultados obtenidos por ambos algoritmos para 10 rutas propuestas.

Prueba N°	Origen	Destino	Ruta más corta (Búsqueda por costo uniforme-UCS)			Ruta con menos estaciones (Búsqueda por amplitud-BFS)			Diferencia entre distancia recorrida [m]	Diferencia entre n° de estaciones	Diferencia entre tiempo de procesamiento [ms]
			Distancia [m]	*N° de estaciones*	*Tiempo de procesamiento [ms]*	*Distancia [m]*	*N° de estaciones*	*Tiempo procesamiento [ms]*			
1	Acceso 14 Sur (12)	DAE (1)	1544	7	283	1727	5	42	183	2	241
2	DAE (1)	Electrónica (8)	1197	5	258	1380	3	47	183	2	211
3	Ingenierías (8)	DAE (1)	1302	6	322	1485	4	59	183	2	263
4	Computación (11)	Cultura Física (18)	1365	6	267	2070	6	70	705	0	197
5	Acceso 18 Sur (13)	Seminarios (2)	1353	4	364	2124	4	62	771	0	302
6	Computación (11)	DAE (1)	1626	7	268	1809	5	46	183	2	222
7	Seminario (2)	Acceso 14 Sur (12)	1578	6	292	2349	6	85	771	0	207
8	Computación (11)	Derecho (16)	744	4	136	744	4	46	0	0	90
9	Terminal STU (5)	Cultura Física (18)	1181	3	345	1181	3	47	0	0	298
10	Ingenierías (8)	DGIE (15)	704	4	247	704	4	65	0	0	182

Tabla 2 Valoración de usuarios.

Estudiante	Efectividad	Eficiencia	Satisfacción	Promedio
1	10	10	10	10
2	10	10	9	9.67
3	9	10	9	9.33
4	10	10	10	10.00
5	9	9	8	8.67
6	10	10	9	9.67
7	9	10	10	9.67
8	9	10	10	9.67
9	9	9	9	9.00
10	9	10	10	9.67
Calificación promedio de App				9.53

En la figura 7 se observa el ejemplo 1, la aplicación presenta la ruta sugerida, la distancia y el tiempo de procesamiento para cada uno de los algoritmos.

Figura 7 Resultados de la prueba número 1.

Por otro lado, se pidió la evaluación de la aplicación a 10 estudiantes de nuevo ingreso, después del uso de la aplicación por una semana, tomando en cuenta 3 aspectos principales:

- Efectividad: ¿La búsqueda de la ruta más corta se realiza con éxito?
- Eficiencia: ¿El tiempo de ejecución es el idóneo?
- Satisfacción: ¿La aplicación agrada al usuario?

En base a los aspectos anteriores se demandó calificar la aplicación en escala del 1 al 10 para cada una de las 3 características mencionadas, siendo 1 el incumplimiento de dicha propiedad y 10 la satisfacción total del usuario en dicho aspecto.

Se puede decir que se tuvo una buena aprobación por parte de los estudiantes. Tuvimos algunas recomendaciones destacando que sería idóneo que se proporcionara esta aplicación móvil al momento de realizar la inscripción al servicio LOBOBICI.

4. Discusión

A partir de los datos recopilados en la tabla 1 y 2 se observa que cada algoritmo implementado en MIT App Inventor cumple con su cometido. La ruta más corta, dada por el algoritmo de BFS, entrega el resultado sin tomar en cuenta el número de estaciones. La ruta con menos estaciones, dada por el algoritmo de UCS, arroja el resultado sin tener en cuenta la distancia de la ruta.

Se tomarán 3 ejemplos de la tabla 1 para discutir los resultados obtenidos. En primer lugar, se tienen los casos similares a la prueba 1. El origen es la estación Acceso 14 Sur (estación número 12 en la figura 1), el destino es la estación DAE (estación número 1 en la figura 1). Aplicando ambos métodos se obtienen resultados que se diferencian entre sí de manera notable tanto en distancia como en número de estaciones.

También existen los casos parecidos al de la prueba número 4. Se desea ir de la estación Computación (11) a la estación Cultura Física (18). Las rutas entregadas por los algoritmos si bien son distintas, tienen el mismo número de estaciones, la

diferencia se remarca en la distancia recorrida para cada una. Esto da a entender que aunque una ruta tenga el mínimo de estaciones, no asegura que la distancia sea menor; de manera parecida, el obtener la ruta con la distancia más corta no asegura que el número de estaciones sea el mínimo, como se observa en el ejemplo anterior.

Finalmente se toma el ejemplo de la prueba 9 que busca la ruta entre Terminal STU (5) y Cultura Física (18). Los casos semejantes a éste entregan resultados idénticos para ambos algoritmos, es decir, la ruta propuesta es la solución óptima sin importar si el criterio es la distancia o el número de estaciones. Estas soluciones se dan especialmente cuando las estaciones de origen y de destino se encuentran cercanas.

5. Conclusiones

Los métodos evaluados en este trabajo presentan características similares, de hecho, el algoritmo de búsqueda por amplitud es un caso particular del algoritmo de costo uniforme donde todos sus nodos son iguales. A pesar de ello, la búsqueda por amplitud tuvo mejor desempeño ya que la propiedad principal a evaluar es la distancia que se debe recorrer y este algoritmo tiende a arrojar la menor, esto se refleja en menos consumo de memoria y tiempo de procesamiento en el dispositivo móvil.

De nuestro estudio, podemos destacar la resolución de un problema cotidiano con la ayuda de algoritmos desarrollados mucho tiempo atrás, brindándole al universitario una herramienta eficaz que evita pérdidas de tiempo en su rutina, aunado a su portabilidad y el bajo consumo de recursos en el celular. Del mismo modo nos permite observar el comportamiento de cada método en una aplicación móvil que no había sido evaluada de esta forma hasta ahora en dispositivos de esta índole.

El siguiente paso es desarrollar una aplicación en la que se resuelva un problema de mayor dificultad con el objetivo de comparar más algoritmos de búsquedas no informadas e informadas. Solo así se podrá brindar un mayor panorama para que los interesados puedan optar por una u otra opción.

6. Bibliografía y Referencias

[1] Betali, J. Lecture Notes on Congnitive Science and Search Algorithms. University of California, San Diego. Noviembre, 1999.

[2] Brooks, R. Intelligence without representation. Artificial Intelligence, Volume 47. Enero, 1991.

[3] Chandel, A., & Sood, M., Searching and optimization techniques in Artificial Intelligence: A comparative study & complexity analysis. International Journal of Advanced Reserch in Computer Engineering and Technology. Marzo, 2014.

[4] Chiong, R., & Hadi, J., & Japutra, W., A comparative study on informed and uninformed search for intelligent travel planning in Borneo Island. Sarawak, Malasya. 2008.

[5] Gallo, G., & Pallottino, S., Shortest path methods: a unified approach. Mathematical Programming Study. 1986.

[6] Guerriero, F., & Musmanno, R., Label correcting methods to solve multicriteria shortest path problems. Journal of Optimization Theory and Applications. 2001.

[7] Korf, R. E. Depth-first iterative-deepening, An optimal admissible tree search. Artificial Intelligence. 1985.

[8] Korf, R. E., Scientific paper on Artificial Intelligence Search Algorithms. University of California. Junio, 1999.

[9] Kumar, V., & Ramesh, K., & Rao, V., Parallel Best-First Search of State-Space Graphs: A summary of results. Vol. 88, 1988.

[10] Kuruvilla, M., & Mujahid, T., & Mohana, R., Experimental comparison of uninformed and heuristic AI Algorithms for N puzzle solution. Swinburne University of Technology. Kuching, Malasya. Enero, 2014.

[11] Serrano, A. Inteligencia Artificial: Fundamentos, prácticas y aplicaciones. RC Libros, 2012.