ANTON, J., BRAVO, J., ARCILA, J. y TUESTA, V. Planificador de rutas para recojo de desechos sólidos utilizando el algoritmo de DIJKSTRA. Rev. INGENIERÍA: Ciencia, Tecnología e Innovación. Agosto-Diciembre 2021. Vol. 8/No. 1, pp. 92-99. ISSN: 2313-1926, versión electrónica.

PLANIFICADOR DE RUTAS PARA RECOJO DE DESECHOS SÓLIDOS UTILIZANDO EL ALGORITMO DE DIJKSTRA

PLANNER OF ROUTES FOR COLLECTING SOLID WASTE USING THE DIJKSTRA ALGORITHM

Juan Manuel Anton Bernal[1a]
Jaime Arturo Bravo Ruíz[1b]
Juan Carlos Arcila Díaz[1c]
Victor Alexci Tuesta Monteza[1d]

Fecha de recepción : 09/06/2021
Fecha de aprobación : 23/09/2021

Resumen

En esta investigación se desarrolló un planificador de rutas para el recojo de desechos sólidos ubicados en diferentes puntos de acción (focos infecciosos) en el distrito de Chiclayo, departamento de Lambayeque en Perú; este planificador permite brindar al operario del vehículo la ruta más corta que debe seguir para poder realizar la recolección de los desechos sólidos. Para el desarrollo del planificador se implementó el algoritmo de Dijkstra con el objetivo de optimizar las rutas de recolección de desechos sólidos en un sistema informático, tomando como fuente de datos un grafo dirigido ponderado mediante una matriz de adyacencia, ya que el recorrido lo realizará un vehículo terrestre, el grafo representa el mapa del distrito de Chiclayo, donde se ingresaron los focos infecciosos a recorrer y el punto de partida tomando en cuenta la dirección de las calles y/o avenidas del distrito de Chiclayo, para la obtención y visualización de mapas geográficos se empleó el API V3 de Google Maps. En los resultados obtenidos durante la simulación del planificador de rutas utilizando el algoritmo de Dijkstra se observa una tendencia de mejora en el tiempo promedio de ejecución del algoritmo a partir de 20 focos infecciosos en adelante.

Palabras clave: *Optimización de rutas, Localización, Grafo, Matriz de Adyacencia.*

Abstract

In this research, a route planner was developed for the collection of solid waste located at different points of action (infectious foci) in the district of Chiclayo, department of Lambayeque in Peru; This planner allows the operator of the vehicle to take the shortest route to follow to collect solid waste. For the development of the planner, the Dijkstra algorithm was implemented in order to optimize the solid waste collection routes in a computer system, taking as a data source a directed graph weighted by an adjacency matrix, since the route will be carried out by a terrestrial vehicle, the graph represents the map of the district of Chiclayo where the infectious centers to be traveled were entered and the starting point taking into account the direction of the streets and / or avenues of the district of Chiclayo, for the obtaining and visualization of maps geographic maps were used Google Maps API V3. In the results obtained during the simulation of the route planner using the Dijkstra algorithm, an improvement trend is observed in the average execution time of the algorithm from 20 infectious foci onwards.

[1] Escuela Profesional de Ingeniería de Sistemas, Facultad de Ingeniería Arquitectura y Urbanismo, Universidad Señor de Sipán, Pimentel-Chiclayo, Perú

[a] Ingeniero de Sistemas, antonbernal@crece.uss.edu.pe

[b] Master Economie Gestion Communication, Ingeniero en Computación e Informática, jaimebravo@crece.uss.edu.pe, https://orcid.org/0000-0003-1929-3969

[c] Ingeniero de Sistemas, diarcilaju@crece.uss.edu.pe, https://orcid.org/0000-0002-7788-951X

[d] Magister en Administración de Negocios, Ingeniero de Sistemas, vtuesta@crece.uss.edu.pe, https://orcid.org/0000-0002-5913-990X

Keywords: *Route optimization, Location, Graph, Adjacency Matrix.*

1. Introducción

A nivel mundial el tema del cálculo de la ruta o camino más corto juega un papel muy importante en las aplicaciones de red de carreteras como en casos de emergencia, sistema asistente del conductor, sistemas modernos de navegación para automóviles, entre otros, una solución rápida al problema del cálculo del camino más corto es utilizar el algoritmo de Dijkstra que está diseñado para calcular la ruta más corta (Makariye, 2017).

En la actualidad existen diversos problemas presentados durante la recolección de Residuos Sólidos Municipales (RSM), entre estos problemas tenemos:

- Las rutas existentes de recolección no han pasado por un proceso de optimización.
- Los operarios de los camiones no están capacitados.
- Los camiones deben recorrer grandes distancias algunas de ellas innecesarias.

Las distancias innecesarias recorridas por el camión suceden porque este pasa varias veces por la misma ruta debido a que las rutas de recolección no están optimizadas (Cusco Tenesaca & Picón Aguirre, 2015).
En el distrito de Chiclayo, departamento de Lambayeque en Perú se presentan estos mismos problemas, los vehículos que realizan el servicio de recojo de desechos sólidos cumplen con turnos y recorridos diferentes, sin embargo, el servicio de recolección que brindan diariamente se realiza de manera no planificada, por lo que algunas veces se recorren rutas innecesarias (Municipalidad Provincial de Chiclayo, 2013).

En la actualidad diferentes autores han enfrentado problemas relacionados a la planificación de rutas en escenarios diferentes, como se menciona en (Zhang, Chen & Li, 2015) que para empresas grandes, donde sus vehículos recorren grandes distancias, la optimización de rutas es importante en cuestión de logística, es por ello que implementaron el algoritmo de dijkstra para resolver el problema de la ruta más corta, utilizaron el método del mapa etiquetado para visualizar el estado de cada iteración (Xin, Yan, & Taoying, 2015). En el proyecto de investigación (la Cruz et al., 2016), utilizaron los algoritmos Dijkstra, Bellman-Ford y Floyd-Warshall para implementar un mecanismo de búsqueda de rutas más cortas, con el objetivo de proveer al cliente de un supermercado la ruta más corta para comprar sus artículos.

En la presente investigación se desarrolló un planificador de rutas para el recojo de desechos sólidos ubicados en diferentes puntos de acción (focos infecciosos) en el distrito de Chiclayo, que brindará al operario del vehículo la ruta más corta que debe seguir para poder realizar la recolección de los desechos sólidos. Para el desarrollo del planificador se implementó el algoritmo de Dijkstra con el objetivo de optimizar las rutas de recolección de desechos sólidos en un sistema informático, tomando como fuente de datos un grafo dirigido ponderado que representará el mapa del distrito de Chiclayo en el cual se ingresaron los focos infecciosos a recorrer y el punto de partida tomando en cuenta la dirección de las calles y/o avenidas.

El método propuesto consiste en obtener los puntos georreferenciados recopilando información del servidor de aplicaciones de mapas Google Maps, para luego convertir dicha información en un grafo tomando en cuenta la restricción de la matriz de adyacencia que representa el grafo dirigido ponderado. Posteriormente se aplicó el algoritmo de Dijkstra tomando como fuente de datos el grafo dirigido, el punto de partida y los puntos de acción (focos infecciosos), el algoritmo debe realizar el cálculo del vértice adyacente varias veces hasta hallar la mejor ruta es decir la ruta más corta (Xin et al., 2015) (Rosyidi, Pradityo, Gunawan, & Sari, 2014); se diseñó un sistema informático en el lenguaje de programación PHP donde se implementó el algoritmo de Dijkstra, para la obtención y visualización de mapas se utilizó el API V3 de Google Maps, se simularon las pruebas en este sistema informático tomando como indicadores el tiempo de ejecución y costo de la trayectoria.

2. Materiales y métodos

A. Materiales

Para el desarrollo del sistema informático se utilizó:

- El IDE Visual Studio Code V 1.18.1.
- Lenguaje de Programación PHP V 7.1.10.
- Framework VueJS 2.5.1.

Se implementó el API V3 de Google Maps para poder acceder a los marcadores de los mapas de Google, permitir ubicar y posicionar un elemento sobre el mapa del sistema ya sea el punto de partida o los puntos de acción, así como también la función polígona que consiste en dibujar una recta de acuerdo con un vector de coordenadas.

Se utilizaron dos computadores con procesadores CoreI7 y CoreI5 para realizar las pruebas y verificar los resultados obtenidos del algoritmo de Dijkstra sobre el Planificador de rutas:

PC1: Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz, 12.0 GB RAM, Sistema operativo Windows 8.1.

PC2: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 6.00 GB RAM, Sistema operativo Windows 10.

B. Método

El método propuesto en esta investigación se basa en (Alican, Gazihan, & Efendi, 2017), donde se realiza el mapeo de la red de transporte de la ciudad de Izmir (Turquía) que representa el conjunto de datos para aplicar después el algoritmo de Dijkstra y planificar las rutas de transporte público.

En la figura 1 se detalla el planificador de rutas para el recojo de los desechos sólidos utilizando el algoritmo de Dijkstra, tomando en cuenta el punto de partida, los puntos de acción y la dirección de las calles.

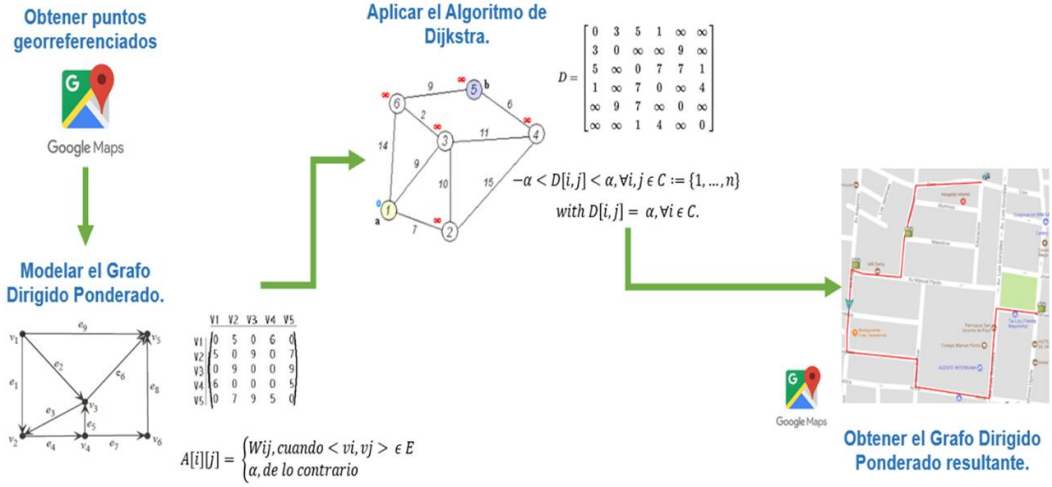

Figura 1. Esquema del método propuesto.
Fuente: Elaboración propia.

Preparar el conjunto de datos

En primer lugar, se preparó el conjunto de datos compuesto por los puntos georreferenciados mediante el servidor de aplicaciones de mapas Google Maps, una vez recopilada la información

se procede a definir la manera en que se representarán los puntos de acción en el grafo siguiendo 3 pasos:

- Poner en marcha el problema: Se expresa el problema presentado a resolver.
- Modelado del problema: Se determinan la manera en que se expresan los datos (vértices y aristas) del grafo dirigido.
- Solución del problema: Se modela el grafo para visualizar como se presentará.

Después de realizar los pasos anteriores se define la manera en que se representarán los datos en el grafo, tales como la manera en que se representarán los vértices (v1, v2, v3…) y las aristas (e1, e2, e3…) (Xin et al., 2015). Luego se construye la matriz de adyacencia tomando en cuenta las restricciones de la matriz sobre el grafo para poder construir el grafo dirigido, como se define en la ecuación (1).

$$\mathrm{A[i][j]} = \begin{cases} w_{ij}, cuando\ \langle vi, vj \rangle\ \in\ E \\ \propto, de\ lo\ contrario \end{cases} \quad (1)$$

La matriz de adyacencia deberá cumplir dos condiciones:

- Existe un arco siempre y cuando $\langle vi, vj \rangle$ estén conectados por una arista y sea accesible.
- De lo contrario, no se considera adyacente por lo tanto se interpretará como $\propto$.

En la figura 2 se muestra la red de calles y/o avenidas del distrito de Chiclayo -Lambayeque.

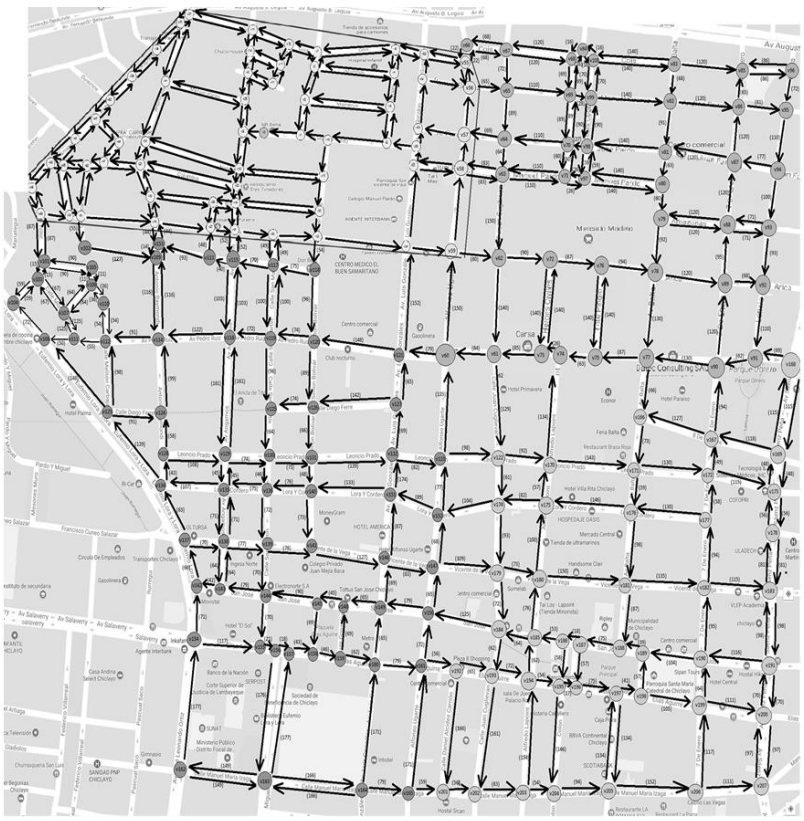

Figura 2. Red de calles y/o avenidas del distrito de Chiclayo.
Fuente: Elaboración propia.

Posteriormente al modelado del grafo dirigido ponderado se procede a implementar el algoritmo de Dijkstra para poder calcular la ruta de menor costo.

El Algoritmo de Dijkstra (Ge Xiaoxue, 2017) es un algoritmo de búsqueda que se encarga de calcular el problema de la ruta más corta entre dos puntos (Rosyidi et al., 2014) (Ma et al., 2016).

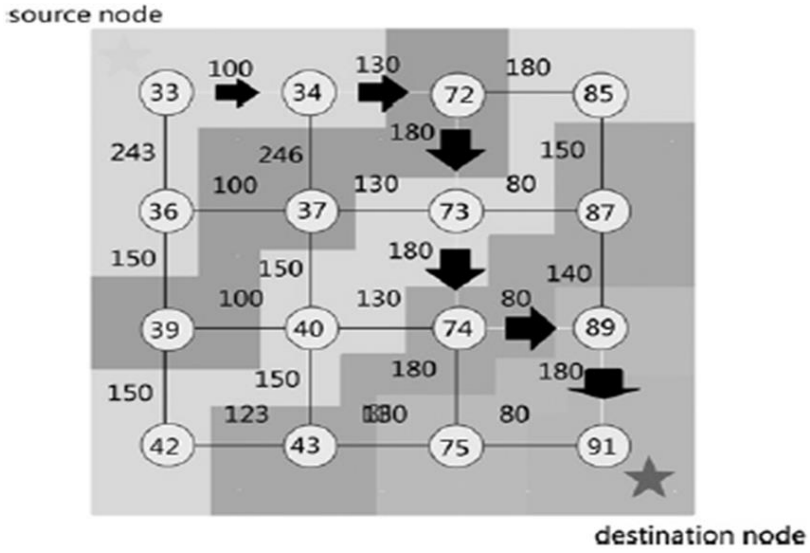

Figura 3. Proceso de enrutamiento óptimo.
Fuente: (Chuang, Jianping, & Yu, 2014).

El algoritmo de Dijkstra si bien solo necesita como entrada las distancias entre nodos debe tomar en cuenta lo siguiente:

- n>1, donde n representa el conjunto de nodos del grafo.
- Debe tener una matriz de adyacencia que represente la distancia entre nodos que existen en el grafo.

Lo que significa que el algoritmo de Dijkstra debe verificar que la matriz de adyacencia cumpla lo que se define en la ecuación (2).

$$-\propto < D[i,j] < \propto, \forall i,j \in C \coloneqq \{1,\dots,n\} \quad (2)$$
$$\text{Con } D[i,j] = \propto, \forall i \in C$$

- La matriz de adyacencia debe ser mayor que $-\propto$ y menor que $\propto$ para todo *(i, j)* perteneciente al conjunto de vértices del grafo.
- Con la matriz de adyacencia igual que $\propto$, se interpreta como una indicación de que no hay un enlace directo del nodo *i* al nodo *j*, para toda *i* perteneciente al conjunto *C*, el conjunto *C* representa el grafo.

El diagrama del sistema planificador de rutas (Risald, Merino, & Suyoto, 2017) se aprecia en la figura 4.

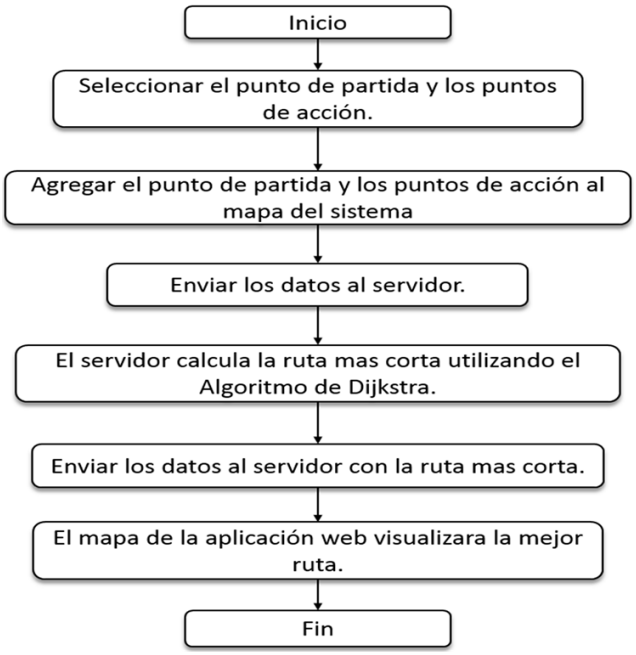

Figura 4. Diagrama de flujo del sistema.
Fuente: Elaboración propia adaptado de (Risald, Merino, & Suyoto, 2017).

3. Resultados

Las simulaciones realizadas durante la aplicación del algoritmo de Dijkstra sobre el planificador de rutas son para 01, 05, 10, 15, 20, 25 y 30 puntos de acción, donde los puntos de acción son diferentes en cada caso (Chuang, Jianping, & Yu, 2014).

Un ejemplo de la simulación para 5 puntos de acción se muestra en la figura 5.

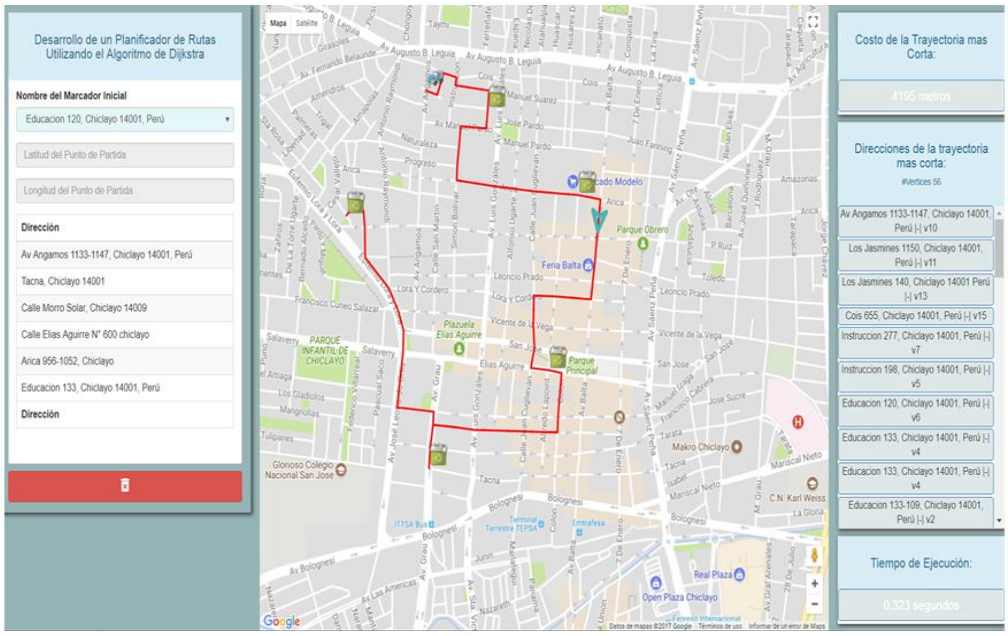

Figura 5. Ruta más corta del algoritmo de Dijkstra para 05 puntos de acción.
Fuente: Elaboración propia.

Los indicadores para observar el comportamiento del algoritmo son los siguientes:

1. Costo: Es la distancia total procesada por el algoritmo de Dijkstra, que sería la distancia más corta necesaria para realizar el recojo de desechos de todos los puntos de acción seleccionados.
2. Tiempo de ejecución: Es el tiempo total que se ejecuta el algoritmo para encontrar la ruta más corta.

En la tabla 1 se detalla los resultados obtenidos de las simulaciones considerando el indicador tiempo, en la tabla 2 se muestra los resultados obtenidos considerando el indicador costo total, siendo el mismo costo obtenido en las pruebas realizadas en ambas computadoras.

Tabla 1
Tiempo de ejecución de la trayectoria

Puntos de acción	Tiempo de ejecución PC1 (s)	Tiempo de ejecución PC1 (s)
1	0,052	0,048
5	0,221	0,219
10	0,529	0,535
15	1,030	1,038
20	1,518	1,663
25	2,282	2,680
30	3,158	4,003

Fuente: Elaboración propia

Tabla 2
Costo total de la trayectoria

Puntos de acción	Costo de trayectoria PC1 – PC2 (m)
1	799m
5	1922m
10	2949m
15	4572m
20	5133m
25	5044m
30	5462m

Fuente: Elaboración propia

4. Discusión

Respecto al tiempo de ejecución del algoritmo de Dijkstra sobre el planificador de rutas se observa que:

Para el computador con procesador Core i7 se obtiene que el algoritmo de Dijkstra para 1 foco infeccioso tiene un promedio de tiempo de ejecución de 0.052s, para 5 focos infecciosos de 0.221s, para 10 focos infecciosos de 0.529s, para 15 focos infecciosos de 1.030s, para 20 focos infecciosos de 1.518s, para 25 focos infecciosos de 2.282s y para 30 focos infecciosos se tiene un promedio de tiempo de ejecución de 3.158s.

Para el computador con procesador Core i5 se obtiene que el algoritmo de Dijkstra para 1 foco infeccioso tiene un promedio de tiempo de ejecución de 0.048s, para 5 focos infecciosos de 0.219s, para 10 focos infecciosos de 0.535s, para 15 focos infecciosos de 1.518s, para 20 focos infecciosos de 1.663s, para 25 focos infecciosos de 2.680s y para 30 focos infecciosos se tiene un promedio de tiempo de ejecución de 4.003s.

Respecto al Costo Total de la trayectoria utilizando el algoritmo de Dijkstra sobre el planificador de rutas se observa que:

Para el computador tanto con procesador Core i7 como Core i5 se obtiene que el algoritmo de Dijkstra para 1 foco infeccioso tiene un costo total de 799m; para 5 focos infecciosos de 1922m; para 10 focos infecciosos de 2949m; para 15 focos infecciosos de 4572m; para 20 focos infecciosos de 5133m; para 25 focos infecciosos de 5044m y para 30 focos infecciosos se tiene un Costo Total de 5462m.

5. Conclusiones

Se observó que el algoritmo de Dijkstra sobre un planificador de rutas con un computador con procesador Core i7 a partir de 20 focos infecciosos, comienza a mostrar una tendencia de mejora en el tiempo de ejecución respecto al computador con procesador Core i5, sin embargo, el costo total de la trayectoria en cada caso es invariable por lo que el costo de la trayectoria es indiferente del procesador en el que se ejecute, siendo así confiable el planificador de rutas.

6. Referencias

Alican, B., Gazihan, A., & Efendi, N. (2017). Public Transport Route Planning: Modified Dijkstra Algorithm. *2nd International Conference on Computer Science and Engineering*, 502–505.

Chuang, R., Jianping, L., & Yu, W. (2014). Map navigation system based on optimal Dijkstra Algorithm. *CCIS - Proceedings of 2014 IEEE 3rd International Conference on Cloud Computed and Intelligence Systems*, 559–564.

Cusco Tenesaca, J. W., & Picón Aguirre, K. (2015). Optimización de rutas de recolección de desechos sólidos Domiciliarios mediante uso de herramienta SIG. *Revista de Teledetección - Asociación Española de Teledetección*.

Ge Xiaoxue, Y. (2017). Optimal Route Planning of Parking Lot Based on Dijkstra Algorithm. *2017 International Conference on Robots Y Intelligent System Optimal*, 221–224.

la Cruz, J., Magwili, G., Mundo, J. P., Gregorio, G., Lamoca, M., & Jasmin, V. (2016). Items-mapping and Route Optimization in a Grocery Store using Dijkstra ' s , Bellman-Ford and Floyd- Warshall Algorithms. *School of EECE, Mapua Institute of Technology*, 244–247.

Ma, B., Feng, Y., Jia, X., Wang, G., Zhang, J., Li, R., & Shi, F. (2016). Vehicle Routing in Urban areas based on the Oil Comsumption Weight-Dijkstra Algorithm. *IET Intelligent Transport Systems*, 495–502.

Makariye, N. (2017). Toward Shortest Path Computation using Dijkstra Algorithm. *International Conference on IoT and Application (ICIOT)*, 1–3.

Municipalidad Provincial de Chiclayo. (2013). Plan integral de gestión ambiental de residuos sólidos de la provincia de Chiclayo, Departamento de Lambayeque, 346.

Risald, Merino, A., & Suyoto. (2017). Best routes selection using Dijkstra and Floyd-Warshall algorithm. *2017 11th International Conference on Information Y Communication Technology and System (ICTS)*, 155–158.

Rosyidi, L., Pradityo, H., Gunawan, D., & Sari, R. (2014). Timebase dynamic weight for Dijkstra Algorithm implementation in route planning software. *Proceedings of 2014 International Conference on Intelligent Green Building and Smart Grid, IGBSG 2014*.

Xin, Z., Yan, C., & Taoying, L. (2015). Optimization of logistics route based on Dijkstra. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, 313–316.

Zhang , X., Chen, Y., & Li, T. (2015). Optimization of logistics route based on Dijkstra. Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on. Beijing. doi:10.1109/ICSESS.2015.7339063