Universidad de Lima

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

ALGORITMO MEJORADO BASADO EN KRUSKAL Y PRIM QUE RESUELVE EL PROBLEMA DEL REPARTIDOR ORIENTADO A TIENDAS CON ALMACÉN CENTRALIZADO USANDO DATOS DEL API DIRECTIONS DE GOOGLE

Tesis para optar el Título Profesional de Ingeniero de Sistemas

Franco Mauricio Ruiz Landers

Código 20121143

Asesor

Hernan Alejandro Quintana Cruz

Lima – Perú

Julio de 2023

ALGORITMO MEJORADO BASADO EN KRUSKAL Y PRIM QUE RESUELVE EL PROBLEMA DEL REPARTIDOR ORIENTADO A TIENDAS CON ALMACÉN CENTRALIZADO USANDO DATOS DEL API DIRECTIONS DE GOOGLE

Franco Mauricio Ruiz Landers
20121143@aloe.ulima.edu.pe
Universidad de Lima

RESUMEN

La presente investigación tiene como principal objetivo resolver un tipo de problema de enrutamiento de vehículos. El problema se enfoca en couriers que realicen entregas a varios destinos. Se busca aplicar la teoría de grafos, tomando los destinos de las entregas como los nodos y el tiempo o la distancia como los vértices. Para resolverlo, se debe aplicar una forma de convertir ese grafo en un Árbol de Expansión Mínima. Por lo tanto, se realiza una comparación entre el algoritmo de Prim y una modificación del algoritmo de Kruskal ya que ambos permiten generar un Árbol de Expansión Mínima, el cual nos permite reducir el costo total de los vértices. La implementación de la solución consiste en una aplicación Node.js en la que se ingresar direcciones en coordenadas como muestras a un motor que permite crear la matriz de costos de un grafo, en el cual, los nodos equivalen a los destinos y el peso de los vértices, a la distancia entre cada destino. Dentro de este procesamiento se hace uso del recurso abierto en línea 'Directions' de Google, orientado al cálculo de rutas entre dos direcciones, del cual se registran las variables distancia y duración de los resultados. De esta manera, se continúa ingresando la matriz de costos como variable de entrada a cada algoritmo. Finalmente, se genera la lista de destinos donde el courier debe entregar pedidos en el orden óptimo. Para entender el impacto, se compara el costo total del orden óptimo generado por Kruskal y el orden en el que fue entregado originalmente. En cuatro de cada cinco muestras se encontró una mejora de 11% en promedio de los pesos totales del grafo que representa la matriz de costos, lo que nos indica que los distribuidores sí pueden reducir los costos de entrega.

PALABRAS CLAVE: Problema de Enrutamiento de Vehículos – Algoritmo de Prim – Algoritmo de Kruskal – Árbol de Mínima Expansión – Algoritmos codiciosos - Ciencias de la computación

ABSTRACT

The main objective of this research is solving a variant of VRP (Vehicle Routing Problem) focused on couriers that delivery to many places at once. Following graph theory, nodes and vertices are now destinations and time/distance. In order to solve this, we must apply a way to turn the graph into a Minimum Spanning Tree. Hence the comparison between algorithm of Prim and a modification of Kruskal's due to both achieve to generate Minimum Spanning Tree, which lets us reduce total costs of vertices. Implementation of solution consists in a HTML web application programmed using Javascript, which works calculating total time and distance cost of a list of destinations in order to generate graph's cost matrix in which nodes represent destinations as coordinates and vertices the route from point A to B. Within computation, Google's online free resource 'Directions' is used to calculate time and distance between each par destination as input. Finally, it outputs an optimal order destination list for the courier. To address impact, there's a comparison between total time and distance cost of optimal order created by Kruskal or original order taken by courier. On four out of five samples, there was found an improvement of 11% on average, which means that distributors can reduce costs of delivery.

KEYWORDS: *Vehicle Routing Problem - Prim's Algorithm - Kruskal's Algorithm – Minimal Expansion Tree - Greedy Algorithms - Computer science*

INTRODUCCIÓN

Los 'couriers' son aquellos que usan normalmente una moto para realizar envíos a domicilio, oficina, etc. (Gevaers, 2014). Son contratados por empresas distribuidoras como también por vendedores por internet. La forma en la que reciben remuneración puede ser por servicio o por el día. Es posible conocer el orden óptimo para completar los destinos (Lozano, 2018; Kin et al., 2016) que priorice la distancia o la duración del viaje según el caso. De esta manera se puede reducir los costos de envíos o maximizar el volumen de envíos, respectivamente. (Chen, 2017; Hübner et al., 2016).

El coordinador de envíos debe otorgar al courier la lista de destinos a los que debe entregar los pedidos. El orden de entrega puede ser definido por el distribuidor o por el Courier, pero la decisión para elegir el orden es subjetiva y depende de la experiencia de esa persona. En el caso de que se contrate por día, semana o mes y los destinos son cercanos, el courier debe priorizar duración sobre distancia del viaje (Bermudez, 2019); porque el objetivo del coordinador es que el courier regrese para enviar más productos. Por lo tanto, en este caso se hará uso de la variable duración. En el caso de que se contrate por envío, el costo está directamente relacionado a la distancia del viaje; por lo tanto, la prioridad del vendedor es minimizar los km recorridos y la variable a usar será la de distancia. (Valiente Bermudez y Hernandez Velasco, 2019).

Esta investigación tiene la importancia de ofrecer al distribuidor una forma para planificar los pedidos que determine orden en el que debe entregarlos el courier buscando reducir los costos totales en términos de la distancia y/o duración recorrida, lo que puede significar reducción de costos de operación (distancia) o mayor capacidad de envío (duración). A esto se le denomina el problema de enrutamiento de vehículos (Hannan et al., 2017; Chen, 2019).

Para resolver el problema, se modificará un algoritmo voraz ejecutado de forma repetitiva (Cerrone et al., 2018) programado en una función de Javascript y ejecutado en Node.js. Lo que se quiere proponer en el presente estudio es que el orden óptimo sea representado en un grafo. Lo que nos indica la teoría de grafos es que se puede comprender la aplicación de algoritmos de ordenamiento para la creación del árbol de expansión mínima (Inga et al., 2016). Los nodos equivalen a los destinos a los que tiene llegar el courier y los vértices, los costos totales que se generan al viajar de un punto de origen a un punto de destino. El costo total, en términos de distancia o duración, se calcula haciendo consultas al servicio 'Directions' de Google (Prasetyo et al., 2018), el cual recibe como variables de entrada los destinos como coordenadas. Finalmente se ejecuta el algoritmo modificado, que consiste en una reorganización de los resultados de Kruskal (Li et al., 2017) para luego compararlos con Prim (Zeng et al., 2019; Granera et al., 2016).

El objetivo al que se quiere llegar en este estudio es demostrar la factibilidad de usar un algoritmo mejorado basado en Kruskal que indique el orden óptimo que debe seguir un courier en caso tenga más de un destino en contraste con elegir el orden de forma arbitraria. Para lograrlo, será necesario calcular los costos totales del itinerario de entrega original, aplicar una modificación de Kruskal para generar un árbol de expansión mínima, realizar la comparación de los resultados usando el modelo de validación propuesto y, finalmente, indicar el orden óptimo por muestra.

ESTADO DEL ARTE

Algoritmos codiciosos

Para Benavides Larreina (2017), la ruta óptima es representada como una línea de transmisión y su objetivo es evitar la congestión de red considerando las limitaciones en los costos. El autor aplica la teoría de grafos y realiza una comparación entre los algoritmos de Kruskal y Prim. Los vértices del grafo se componen por los costos de operación y mantenimiento. Por ejemplo, los costos de combustible y de desgaste de llantas. Finalmente pudo crear un árbol de expansión mínima que permite eficiencia en la transmisión a pesar de la cantidad de nodos de la red.

Siahaan et al. (2018) buscan medir el rendimiento de los algoritmos Prim y Genético para determinar la ruta optima de un grafo. La metodología empleada es de comparar las pruebas con diferentes posiciones en el mapa, lo cual genera un grafo. Cada nodo del grafo está interconectado para hallar el árbol de expansión mínima; por consiguiente, se creó una matriz de costos de dicho grafo.

'Directions' de Google

Kiraly y Abonyi (2014) propusieron un método para evitar que uno o más paquetes sea pasado por almacenes logísticos innecesariamente. La técnica del cromosoma múltiple permitiría resolver el problema de enrutamiento de vehículos usando el algoritmo genético. Los diferentes costos que ocurren por enviar por camión son calculados usando 'Directions'.

Prasetyo et al. (2018) desarrollaron un sistema que permite a los dueños de las pensiones para los estudiantes universitarios conectarse con los usuarios nuevos. Aplicaron el algoritmo semiverseno en forma de cascada, el cual es similar al algoritmo de Prim, pero que incluye un cálculo trigonométrico que consta de las variables de latitud y longitud y que toma en cuenta la curvatura de la tierra. Los datos de entrada son el resultado de una consulta a 'Directions', la cual recibe las ubicaciones de las pensiones y genera un recorrido del algoritmo. Rahmi et al. (2017) utilizaron 'Directions' de forma similar en una aplicación de salud de un centro médico.

Problema de enrutamiento de vehículos

Hübner et al. (2014) propusieron realizar encuestas a centros de distribución y aplicar análisis de textos. Concluyen en realizar un planeamiento desde el almacén central que tome en cuenta el stock total de productos y variables como ubicación, automatización e integración. De esta manera se generan los grupos de envío.

Kin et al. (2018) hicieron uso del software SYMBIT que sirve para simular escenarios de entrega de 'bundle' que toma en cuenta la descentralización de la distribuidora. El objetivo de la investigación es poner a trabajar la mayor cantidad de couriers. La prueba se realizó entre ocho escenarios, los cuales tienen las variables de porcentaje de ordenes en línea y porcentaje de uso de almacén central. Concluyen en que es más seguro tener dos almacenes centrales; uno para completar las órdenes de la tienda, y otro para completar las órdenes.

Ayu (2015) buscó resolver una variante del Problema de Enrutamiento de Vehículos. Su metodologia consuste en usar técnicas de descomposición para separar los destinos en grupos para luego realizar el cálculo de la 'ruta óptima' usando algoritmos heurísticos. Esta variante se denomina 'Problema de Enrutamiento de Vehículos Asistido' (Capacitated VRP) en la cual la demanda es conocida lo que facilita dividir los destinos en grupos. Luego aplica los métodos 'ahorro Clarke-Wright' y 'el vecino cercano'. Para determinar las rutas de cada grupo, se aplica el 'método de barrido' y de 'vecino cercano' nuevamente. Los resultados indican que esta combinación de metodologías de enrutamiento permite al proveedor reducir el número máximo de couriers necesarios diariamente.

ANTECEDENTES

Árbol de Expansión Mínima

Un árbol de expansión mínima es aquel que tiene como objetivo recorrer los vértices de un grafo que tengan el menor peso posible de cada uno de los vértices de los que ese se compone. Se trata de un subgrafo que tiene la posibilidad de pasar por todos los nodos de un grafo después de conocer los vértices y los pesos de cada uno de ellos, según Inga et al (2016). Además, en algunos casos, cuando hay demasiados registros de datos, puede separar los nodos en grupos. (Grygorash et al., 2006; Coto, 2003)

Algoritmos Codiciosos

Un algoritmo codicioso es aquel que realiza una actividad cíclica y que busca hallar el valor óptimo para cada una de las repeticiones. En la Tabla 1 se presenta el pseudocódigo de los algoritmos de Prim y Kruskal originales

Tabla 1

Pseudocódigo de Prim y Kruskal

Prim (Grafo G)	función Kruskal(G)

Distancia= valorMaximo
//Encolamos, en una cola de prioridad todas las parejas <nodo, distancia> del grafo
por cada u en V[G] hacer
  distancia[u] = INFINITO
  padre[u] = NULL
  Añadir(cola,<u, distancia[u]>)
  distancia[u]=0
	Para cada v en V[G] hacer
	Nuevo conjunto C(v) ← {v}.
	Nuevo heap Q que contiene todas las aristas de G, ordenando por su peso
	Defino un árbol T ← ∅
	// n es el número total de vértices
	// Mientras T tenga menos de n-1 aristas y !Q.vacío() hacer
	(u,v) ← Q.sacarMin()
	// previene ciclos en T. agrega (u,v) si u y v están diferentes componentes en el conjunto.
	Si C(v) ≠ C(u) hacer
	  Agregar arista (v,u) a T
	  Merge C(v) y C(u) en el conjunto
	Responder árbol T

Nota. De *Algoritmos Básicos de Grafos*, por Lecturas en Ciencias de la Computación, 2003

METODOLOGÍA

Para resolver el problema de enrutamiento de vehículos orientado a couriers con muchos destinos, se debe otorgar al courier el orden en el que debe entregar los pedidos tomando las rutas con menor duración o distancia (según el caso). El orden óptimo puede ser hallado usando la teoría de grafos en el cual los nodos son representados por los destinos de los pedidos y los vértices por los costos totales de las rutas indicadas por 'Directions'. Un árbol de expansión mínima nos permite minimizar el peso total de los vértices. Para que un grafo se convierta en un árbol de expansión mínima, se debe ejecutar un algoritmo codicioso (Inga et al, 2016; Grygorash et al, 2006), lo cual será aplicado haciendo uso del algoritmo de Prim y una modificación del algoritmo de Kruskal.

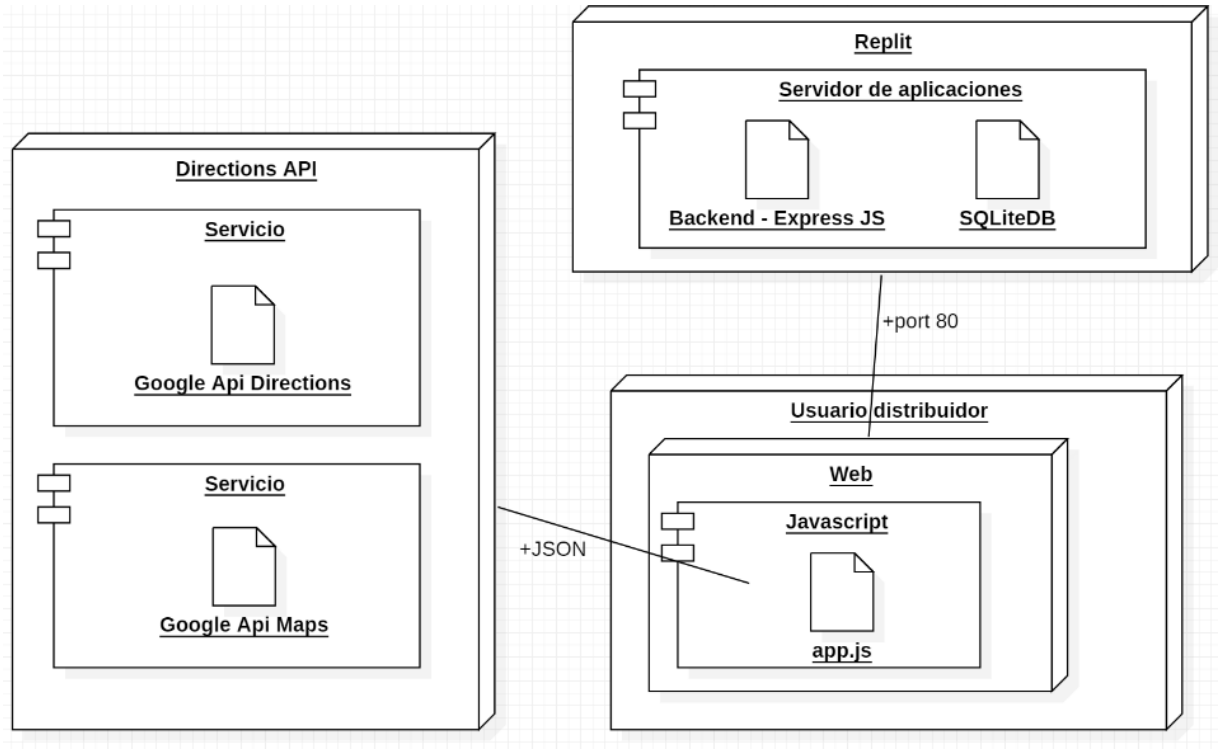
El modelo consiste en calcular el costo total de entregas que se realizaron por un lado y, por otro lado, se ingresan los destinos en una función que se conecta a 'Directions' de forma iterativa para generar una matriz de costos con los resultados de la función 'DirectionsService()'. Esta función recibe un par de destinos y devuelve la distancia total en km recorridos, la duración en segundos, tipo de direcciones (DRIVING, WALKING, BUS, etc.), información sobre las direcciones ('addresses') de inicio y fin, entre otros. Para generar la matriz de costos se debe tomar el valor de distancia o de duración de cada par de destinos de la muestra.

La matriz de costos generada sirve como variable de ingreso al algoritmo mejorado de Kruskal y al algoritmo de Prim. Finalmente se comparan los costos totales del itinerario original y el orden óptimo de que fue definido por Prim o por Kruskal y se otorga el orden en el que el courier debe hacer las entregas.

En la primera parte del experimento, el distribuidor ingresa los destinos a los que debe llegar un Courier para que se realicen consultas a Google. Los algoritmos se ejecutan posteriormente dentro del mismo entorno. Como se puede apreciar en la Figura 2, el servidor de aplicaciones hace uso de las funciones para generar la matriz de costos y para ejecutar los algoritmos. La base de datos consiste en los destinos en el orden en el que fueron compartidos por los distribuidores. La lista original también sirve como variable de ingreso en la función 'ordenOriginal', que otorga el costo total de realizar las entregas en el orden original. El usuario puede interactuar con las funciones de consulta a Google y de ejecución de algoritmos a través de una aplicación Node.js desplegada

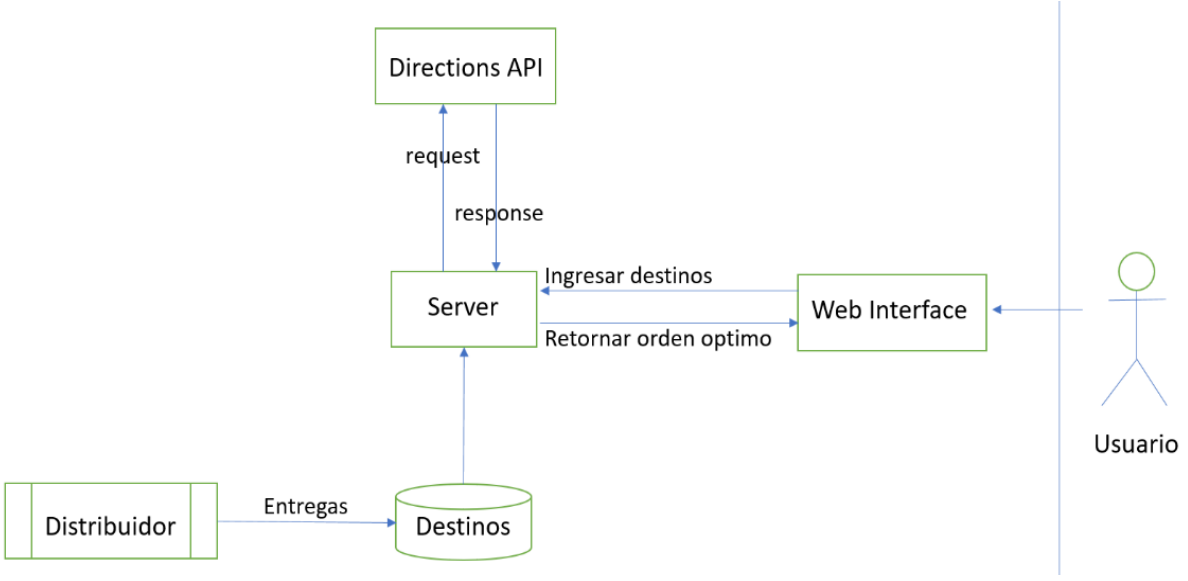
en Replit (Anexo). En esta puede seleccionar los destinos de entrega de una base de datos SQLiteDB, verlos en el mapa y recibir el orden óptimo después de hacer consultas a Directions API de Google.

Figura 2.
Diagrama de despliegue del experimento



En la Figura 3 se relaciona al usuario con la página web que envía los destinos al servidor y recibe el orden optimo.

Figura 3.
Diagrama de arquitectura del experimento



A continuación, se describirán las etapas del experimento:

Realizar consultas con 'Directions' de Google y generar las matrices de costos de los grafos

Figura 4.
Flujo de procesos del experimento

Se tienen las diez listas de los destinos a los que el Courier realizó las entregas en los órdenes en el que fueron entregados. Por cada muestra, se debe calcular el costo total de entrega, ingresar los destinos al algoritmo como variable de entrada y calcular el costo total del orden óptimo indicado por el algoritmo. Finalmente se presentan los resultados del rendimiento del algoritmo y una comparación de los costos totales del orden original del courier y del orden óptimo.

Realizar consultas con 'Directions' de Google y generar las matrices de costos de los grafos

El proceso consiste en ingresar los destinos en forma de coordenadas ("-12.0012121, -78.012128") en una variable *request*; devuelve un objeto JSON con la información general de la ruta entre ambos destinos, del cual se puede obtener la duración y la distancia. Finalmente, quedará registrada una matriz de costos por cada grupo de destinos. El formato de resultados de la consulta se puede ver en la **Tabla 2.**

Tabla 2
Formato de resultados de 'Directions' de Google

Variable	Contenido
distance	text: '2.4 km', value: 2361
duration	text: '9 min', value: 510
end_location	lng: -77.045470, lat: -12.091132
start_location	lng: -77.032111, lat: -12.086065

Ejecutar la modificación de Kruskal y generar un orden óptimo por cada muestra.

Se debe ingresar la matriz de costos en el algoritmo de Prim, el cual genera los resultados de la **Tabla 3**.

Tabla 3
Formato de resultados del algoritmo de Prim

Par de nodos	Costo
De i a j	∑ij
De 0 a 1	76
De 1 a 2	62
De 2 a 4	49
De 4 a 3	98
De 3 a 5	113
Total	398

Se pueden tomar las posiciones de i para generar el orden optimo; no necesita modificación. Se suman las ∑ij para generar el costo total. De la misma forma, se ingresa la matriz de costos en el algoritmo de Kruskal, el cual genera los resultados presentados en la **Tabla 4**:

Tabla 4
Formato de resultados del algoritmo de Kruskal

Par de nodos	Costo
De i a j	∑ij
De 0 a 4	89
De 4 a 1	42
De 1 a 3	50
De 0 a 2	121
De 0 a 5	86
Total	388

No se pueden tomar las posiciones de i para generar un orden porque en algunos casos se repite. En la **Tabla 5** se formula la modificación al algoritmo de Kruskal: Los resultados originales de ejecutar el algoritmo se guardan en un arreglo "resultK".

Tabla 5
Pseudocódigo del algoritmo mejorado basado en Prim y Kruskal

Código programado en Javascript	Explicación por cada línea de comando
Modificacion (arreglo resultK)	
CREAR arreglo grupo, CREAR arreglo par, CREAR arreglo destinos	Se debe ingresar el primer par como valores de inicio.
AGREGAR resultK[0] a grupo	
PARA cada i = 0 del arreglo resultk	Empezamos recorriendo 'resultK'
par ← resultK[i]	Guardamos el par de resultados de Kruskal i.
SI par[0] = 0	Si el primer valor del par i es 0,
AGREGAR par a grupo	el par se registra en caminos.
FINSI	Si no,
PARA cada j = 0 del arreglo grupo	se debe recorrer el arreglo grupo
Destinos ← grupo[j]	Guardamos el grupo de destinos en una variable temporal
PARA cada n = 0 del arreglo destinos	luego recorrer el grupo de destinos
SI destinos[n] = par[0]	Si algún elemento del grupo coincide con el valor inicial
AGREGAR par[1] a destinos	del par de destinos de resultK[i], se añade el segundo
grupo[j] ← destinos	valor a ese grupo
RETORNAR grupo	Retorna el arreglo 'grupo' en el que se dividen los destinos de la muestra.

Nota. De *Repositorio direcciones*, en https://github.com/fruizlanders/direcciones-js

El algoritmo mejorado de Kruskal se trata de un proceso que consiste en recibir los resultados de ejecutar el algoritmo de Kruskal y luego ordenar los arreglos en uno, dos o hasta tres arreglos, en los que se dividen los destinos de la muestra para que sean entregados por varios couriers o por el mismo las veces que se encuentre un arreglo par con el valor '0' en la posición '0'.

En la **Tabla 6** se ejemplifica el proceso de modificación y su resultado.

Tabla 6
Proceso de Modificación de Kruskal y el resultado de cada paso

Paso	Elementos restantes en el arreglo obtenido de ejecutar el algoritmo de Kruskal	Explicación

1	[0, 6] [0, 8] [8, 2] [2, 9] [9, 7] [7, 3] [3, 4] [3, 5] [5, 1]	Se retiran los pares que empiecen con el valor ‘0’, el cual representa el punto de inicio: [0, 6] [0, 8]
2	[8, 2] [2, 9] [9, 7] [7, 3] [3, 4] [3, 5] [5, 1]	Se busca que el valor inicial del siguiente par se encuentre en alguno de los pares que se retiraron en el primer paso y luego se agrega el valor final de ese par en el arreglo donde se encuentra el valor inicial. [0, 6] [0, 8, 2]
3	[2, 9] [9, 7] [7, 3] [3, 4] [3, 5] [5, 1]	[0, 6] [0, 8, 2, 9]
4	[9, 7] [7, 3] [3, 4] [3, 5] [5, 1]	[0, 6] [0, 8, 2, 9, 7]
5	[7, 3] [3, 4] [3, 5] [5, 1]	[0, 6] [0, 8, 2, 9, 7, 3]
6	[3, 4] [3, 5] [5, 1]	[0, 6] [0, 8, 2, 9, 7, 3, 4]
7	[3, 5] [5, 1]	[0, 6] [0, 8, 2, 9, 7, 3, 4, 5, 1]

Descripción de la **Tabla 6**:

1. Se tiene ‘resultK’.
2. Se registran los pares que empiezan en 0.

3. Se revisan de forma iterativa.
4. Se guarda el valor del destino en la lista donde se encuentre el valor de origen

Lo que genera el orden óptimo de la modificación de Kruskal

Comparar el orden indicado por los algoritmos con los definidos por los distribuidores.

El orden óptimo consiste en una lista de destinos en un orden en el cual se toma la menor distancia o el menor tiempo posible. Para medirlo, tomaremos la variable Costo Total, la suma de las distancias o duraciones de cada par de destinos. La comparación de Costos Totales se hará entre el orden óptimo de Prim, de la modificación de Kruskal y el orden en el que fueron entregados los paquetes originalmente.

Se considera el proceso de metodología de Ayu (2015) cuando se aplica la modificación de Kruskal para lograr separar los destinos en grupos, lo cual nos permite evaluar los casos en los que es necesario clusterizar y recibir el orden óptimo que se debe seguir.

EXPERIMENTACIÓN Y RESULTADOS

Esta investigación fue experimental debido a que se utilizaron sujetos de prueba para evaluar si el uso de algoritmos voraces para dividir en grupos y ordenar lograban reducir los costos de operación del courier.

Se utilizaron los itinerarios de entrega de dos distribuidores, dos tipos de casos. El primer caso, se trata de una farmacia que realiza entregas en tres distritos aledaños, un área de diez kilómetros cuadrados. Para este distribuidor, es prioridad que el courier no se demore, por eso la variable a utilizar para ingresar al algoritmo modificado será de duración. Se tomaron los itinerarios realizados por los couriers a lo largo de un mes en el horario de ocho a diez de la mañana. Los destinos se ingresan como matriz en la función que contiene el algoritmo mejorado y el algoritmo de Prim.

En el segundo caso, la tienda en línea que entrega hacia todo Lima Metropolitana sin restricciones de tiempo. Por otro lado, se busca ordenar las entregas para que el courier recorra la menor cantidad de kilómetros. Entonces, para este caso, la variable a ser usada en el algoritmo mejorado de Kruskal y de Prim, será la variable de distancia. Se tomaron los itinerarios realizados por los couriers en diferentes días de la semana. Cada una de estas muestras será ingresada como matriz en los algoritmos presentados.

En ambos casos se busca determinar si aplicando el algoritmo de Prim se mejoran los costos de operación por entrega de pedidos, pero en los casos donde se busca reducir la distancia, se busca también encontrar muestras en las que el algoritmo mejorado de Kruskal divida en grupos los destinos.

Flujo de Implementación:

A. Se ingresan los destinos dentro de la función que permite realizar consultas masivas a 'Directions' y se crea una matriz de costos por cada muestra y por cada variable (duración y distancia). Se realiza la consulta a ‘Directions’ porque nos permite saber la distancia, duración, direcciones, etc. en tiempo real. La ejecución dura aproximadamente un minuto, por lo que no se espera que los cambios en el tráfico afecten el resultado.

B. Luego de crear las matrices, deben ser ingresadas al algoritmo de Prim y al algoritmo mejorado de Kruskal, para obtener los órdenes óptimos según cada algoritmo y según la necesidad del tipo de escenario. Los algoritmos reciben como variable de entrada una matriz, que en realidad es una presentación del grafo.

C. Después de obtener los órdenes óptimos, se pueden ingresar, junto con el orden original de los couriers dentro de la función que nos permite conocer el rendimiento de cada orden, para asegurar que los datos sean homogéneos.

D. Con la información recolectada, se procedió a mostrar los resultados de ejecutar los algoritmos y si mejoran o no los costos después de aplicar la solución. Se debe describir las características de una muestra que elige

a Prim como mejor algoritmo y de otra muestra que divide los destinos en grupos y elige a la modificación de Kruskal para que le indique el orden óptimo.

Los resultados obtenidos se encuentran divididos entre los dos casos (farmacia distribuidora con prioridad en duración de viaje y tienda online con prioridad de distancia de viaje). Si en total son 62 muestras, 22 tuvieron enfoque a la variable Distancia y 40 tuvieron enfoque a la variable duración. Los resultados también se dividen entre las muestras que tomaron a Prim como el orden óptimo, las que aceptaron la modificación de Kruskal para realizar los envíos y aquellas en las que, si bien no se dividió los destinos en grupos con la modificación de Kruskal, el orden óptimo de Prim es el mismo orden que tomó el Courier, por lo tanto, no hay 'mejora' y se excluyen de la prueba.

Los resultados sirven para poder comparar los costos totales del orden original y orden óptimo. En **la Tabla 7**, se puede apreciar que los costos totales del orden óptimo son menores que los costos totales del orden original. En la **Tabla 8**, se presenta la mejora promedio y sus límites inferior y superior.

Tabla 7
Resultados de los costos totales por caso

Indicadores	Duracion (minutos)	Distancia (kilómetros)
Rangos de los costos totales originales	17:31 – 56:32	22 – 55
Rangos de los costos totales del orden óptimo	16:55 – 52:48	20 – 53
Promedio original	33:40	39
Promedio de orden óptimo.	30:46	36

Tabla 8
Mejora porcentual por caso

Indicadores	Duracion (minutos)	Distancia (kilómetros)
Rango	1.76% – 19.43%	1.21% – 23.66%
Promedio	10.65%	11.16%

DISCUSIÓN

Los datos generados y resultados, muestran una tendencia a preferir el algoritmo de Prim sobre la modificación de Kruskal. Realmente no se busca comparar sus resultados, sino combinar las bondades de cada uno para solucionar un tema tan complejo como la logística con almacén centralizado. A diferencia de lo propuesto por Benavides Larreina (2017) y Furqan et al. (2018), que realizan una comparación del rendimiento de Prim y Kruskal (y otros algoritmos codiciosos también), en esta investigación se acepta al algoritmo Prim como el de ordenamiento y la modificación del algoritmo Kruskal nos permite determinar si es necesario separar los destinos en grupos. El algoritmo de Prim propone que un solo repartidor realice las entregas, en cambio Kruskal permite que los envíos se dividan en grupos.

Para medir los resultados y reconocer que: se trata de una solución para el problema de enrutamiento de vehículos, que el algoritmo modificado de Kruskal es tan útil como Prim para generar un árbol de expansión mínima y que los distribuidores no eligen la ruta óptima por defecto, sino que es necesario usar el algoritmo mejorado de Kruskal en conjunto con Prim, se establecerán los siguientes estadísticos:

- **Prueba de hipótesis**

  Se proponen las hipótesis nula y alternativa: 'H0' y 'H1'.

$$H_0: \mu \geq 11$$
$$H_1: \mu < 11$$

Con una significancia de α = 0.05, se calculan el valor crítico y de prueba. Ya que la muestra es mayor a 30, el valor crítico '$Z_{\propto}$' se calcula usando el α = 0.05 y el de prueba 'Z_p' mediante la siguiente fórmula:

$$\frac{\bar{x}-\mu}{\frac{\sigma}{\sqrt{n}}} \qquad (1)$$

Nota. De *Inferencia estadística: pruebas de hipótesis*, 2014

Siendo $\bar{x}$ el promedio de la muestra: 10.85; μ, la media poblacional: 11; σ, la desviación estándar de la muestra: 6.22 y n, el número de repeticiones: 57.

$$Z_p = \frac{\bar{x}-\mu}{\frac{\sigma}{\sqrt{n}}} = \frac{10.85-11}{\frac{6.22}{\sqrt{45}}} = \text{-0.16} \qquad (2)$$

Debido al sentido de la hipótesis alternativa (H_1), el valor crítico ($Z_{\propto}$) se determina ingresando el valor de α en una función en la que resulta el valor de -1.64.

Por consiguiente, debido a que se trata de una prueba de hipótesis en la que la media de la muestra de la hipótesis nula es mayor al valor Z_p es menor al $Z_{\propto}$ se rechaza la hipótesis nula, pero si es mayor, se acepta H_0: μ ⩾11. (Villagómez-Ornelas et al., 2014)

Se puede afirmar que el promedio de mejora porcentual debido a utilizar la modificación del algoritmo de Kruskal y el algoritmo de Prim sí supera el 10.85%

Entonces, se acepta la solución y se puede resolver el problema de enrutamiento de vehículos usando la modificación del algoritmo de Kruskal en conjunto con el algoritmo de Prim.

CONCLUSIONES

Mediante la presente investigación y con la información recopilada a través de pruebas, validaciones y ejecuciones de los algoritmos Prim y la modificación de Kruskal, se pudo comprobar, en primer lugar, que los algoritmos de ordenamiento de grafos resolvieron la prerrogativa propuesta al otorgar un plan para entregar los productos reduciendo los costos, esto debido a que se tratan de algoritmos codiciosos, que buscan resolver todas las posibilidades que haya de recorrer un grafo. Los resultados generados por la modificación de Kruskal, combinados con el orden óptimo otorgado por el algoritmo de Prim tienen un gran impacto en cuanto a mejora de la toma de decisiones.

Los casos en los que se aceptó Prim más veces son los que poseen como variable de decisión la duración. Se acepta que está relacionado con la distancia de los destinos, cuando la muestra tiene destinos que se encuentran muy cerca entre sí. El área promedio de las muestras con enfoque en duración es de 2,5 km por 2,1 km (5,25 km cuadrados), en cambio en el caso de los casos en los que se da importancia a la distancia, se tratan de muestras con un área promedio de 9,6 km por 8,7 km (83,52 km cuadrados). Por ende, es más probable que se acepte Prim fácilmente si los destinos están muy cerca.

En el presente estudio, los algoritmos de Prim y la modificación de Kruskal sirvieron para determinar si es necesario dividir en grupos los destinos. No se intenta comparar el rendimiento de los algoritmos. Sin embargo, tomando la matriz de costos generada por Directions como input en algoritmos exactos, heurísticos, metaheurísticos, etc. y buscar el árbol de expansión mínima, como proponen Mendoza (2017) y Carpente y Casas (2013), se podría realizar comparaciones en los resultados de dichos algoritmos.

El estudio tiene importancia porque busca dar solución a uno de los problemas de mayor importancia en temas de transporte y optimización de costos. Para aprovechar los resultados, conclusiones, hallazgos, etc., se podría

trabajar en una aplicación web responsive o móvil que permita realizar el ordenamiento de destinos de forma automática usando solo el arreglo de destinos como variable de entrada. Cualquier emprendedor podría utilizar este nuevo motor para resolver el problema de enrutamiento de vehículos dentro de su organización sin importar su tamaño y sin tener que realizar ninguna modificación en sus procesos de negocio originales y, en teoría, sin costo de implementación. Una empresa de distribución por aplicativo podría absorber estas técnicas y aplicarlo a su servicio de courier.

Los resultados permiten ampliar los conocimientos en la aplicación de recursos en línea como 'Directions', 'Maps', 'Locations', etc. de 'Google API' ya que se encontró información detallada sobre los destinos, lo cual sirvió para aplicar algoritmos voraces o codiciosos tradicionales de manera experimental sea para casos orientados a reducir distancia o duración. Sin embargo, tambien se pueden considerar otras variables que puedan impactar en la lógica de negocio de cada distribuidora. Algunas de estas variables son 'address' ('start' y 'end') , 'location' ('start' y 'end'), 'steps' (pasos, relacionado al algoritmo de Dijkstra), 'polyline', 'travel_mode' y 'traffic_speed_entry'. 'Travel mode' se entiende como el 'modo de viaje' que, por defecto, es 'DRIVING', pero también se puede aplicar 'WALKING', 'BYCICLE' y otros.

REFERENCIAS

Ayu, K. G., Septivani, N., Xu, S., & Kwan, S. (2015). Optimum Clustering and Routing Model using CVRP Cluster-First, Route-Second in a 3PL Provider. International Conference on Industrial Engineering and Operations Management.

Benavides Larreina, F. P. (2017). *Expansión óptima del sistema de transmisión mediante el algoritmo de Prim*. Quito: Universidad Politécnica Salesiana. url: https://dspace.ups.edu.ec/handle/123456789/14358

Bouamama, Blum y Boukerram (2012). *A population-based iterated greedy algorithm for the minimum weight vertex cover problem.* Applied Soft Computing. doi: 10.1016/j.asoc.2012.02.013

Cardozo, O. D., Gomez, E. L., y Parras, M. A. (2009). *Teoría De Grafos Y Sistemas De Información Geográfica Aplicados Al Transporte Público De Pasajeros En Resistencia*. Revista Transporte y Territorio, 89-111. doi: 10.34096/rtt.i1.223.

Carpente, M. L. y Casas B. (2013) *ALGORITMOS HEURÍSTICOS EN OPTIMIZACION*. Facultad de Matemáticas. Universidad de Santiago de Compostela, Santiago de compostela; url: http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_782.pdf.

Cerrone, Carmine y Cerulli. (2017). *Carousel Greedy: A Generalized Greedy Algorithm with Applications in Optimization. Computers & Operations* Research. 85. 10.1016/j.cor.2017.03.016.

Chen, P. y Chankov, S.M. (2017). *Crowdsourced Delivery for Last-Mile Distribution: An Agent-Based Modelling and Simulation Approach.* Department of Mathematics & Logistics, Jacobs University Bremen. IEEE: 978-1-5386-0948-4/17

Coto, E. (2003). *Data Structures and Algorithms*. Caracas: Universidad Central de Venezuela.

Dagnino, J. (2014). *Inferencia Estadística: Pruebas De Hipótesis*. Rev Chil Anest, 43: 125-128.

De Greef, L. (2017). *Algoritmos Básicos de Grafos*. Lecture 16: Dijkstra's Algorithm (Graphs).

Furqan, M., Mawengkang, H., Sitompul, O. S., Siahaan, Siahaan, M. D., Wanayumini, . . . Sriani. (2018). *A review of Prim and genetic algoritmths in findign and determining routes on connected weighted graphs*. International Journal of Civil Engineering and Technology (IJCIET), 1755–1765.

Gevaers, Van de Voorde, Vanelslander. (2014). Cost Modelling and Simulation of Last-mile Characteristics in an Innovative B2C Supply Chain Environment with Implications on Urban Areas and Cities. University of Antwerp, Department of Transport and Regional Economics, Prinsstraat 13, 2000.

Granera, J. A., Valdivia, V. M., y Blandón Dávila, M. E. (2016). *Aplicación informática KPTS (Kruskal, Prim, Tabu Search).* Revista Científica De FAREM-Estelí, (17), 81–90. Recuperado a partir de https://rcientificaesteli.unan.edu.ni/index.php/RCientifica/article/view/1414

Grygorash, O., Zhou, Y., y Jorgesen, Z. (2006). *Minimum Spanning Tree Based Clustering Algorithms.* Mobile: School of Computer and Information Sciences.

Hannan, M. A., Akhtar, M., Begum, R. A., Basri, H., Hussain, A., y Scavino, E. (2017). *Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm*. Elsevier. doi: 10.1016/j.wasman.2017.10.019

Hernandez Romero, Y., y Galindo Sosa, R. V. (2016). *Modelo de gestión del servicio de transporte UBER*. ¿Quién pierde y quién gana? Espacios Públicos, 157-175.

Hübner, A., Kuhn, H., Wollenburg, J. (2016), *Last mile fulfilment and distribution in omni-channel grocery retailing*, International Journal of Retail & Distribution Management, Vol. 44

Inga, Carrion, Aguila, Garcia, Hincapie y Gonzalez (2016), *Minimal Deployment and Routing Geographic of PMUs on Electrical Power System based on MST Algorithm*, IEEE Lat. Am. Trans., vol. 14, no. 5, pp. 2264–2270. DOI: 10.1109/TLA.2016.7530422.

Kin, B., Ambra, T., Verlinde, S. y Macharis, C. (2016), *Tackling Fragmented Last Mile Deliveries to Nanostores by Utilizing Spare Transportation Capacity—A Simulation Study*, Sustainability 2018, 10, 653; DOI:10.3390/su10030653.

Király y Abonyi. (2015) *Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google Maps API*. Engineering Applications of Artificial Intelligence. Volume 38. Pages 122-130. DOI: 10.1016/j.engappai.2014.10.015.

Li, H., Xia, Q., y Wang, Y. (2017). *Research and Improvement of Kruskal Algorithm*. Journal of Computer and Communications, 63-69. doi:10.4236/jcc.2017.512007

Lozano, N. (2018). *Lo pedís, lo tenés. Caso: Rappi*. Universidad del Salvador

Mendoza, J. (2013) *TRAVELING SALESMAN PROBLEM (TSP) Diseño de Algoritmos Heurísticos y Metaheurísticos eficientes para resolver el Problema del Agente Viajero*. FACULTAD REGIONAL MULTIDISCIPLINARIA DE CHONTALES, Universidad Nacional Autónoma de Nicaragua, Managua; url: repositorio.unan.edu.ni/8779/1/11129.pdf.

Prasetyo, S., Utomo, A. y Hudallah, N. (2018). *Implementation of Google Maps API 3 with Haversine Algorithm in the Development of Geographic Information System Boarding House Finder*. In Proceedings of the 7th Engineering International Conference on Education, Concept and Application on Green Technology (EIC 2018), pages 227-233. DOI: 10.5220/0009008902270233

Rahmi, A., Piarsa, I., y Buana, P. W. (2017*). FinDOctr - Interactive android cloinic geographical information system using firebase and Google Maps API*. International Journal of New Technology and Research (IJNTR), 08-12.

Ramadhan, Z., Siahaan, A. P., y Mesran, M. (2018). *Prim and Floyd-Warshall Comparative Algorithms in Shortest Path Problem*. Medan: ICASI.

Ruales, J. S. (2019). *Rappi: cambiando el ecosistema comercial en América Latina*. Revista Palmas, 275-277.

Siahaan, A. P., Rusiadi, Kan, P. L., Fazira, K. N., Azir, K., y Amir, A. (2018). *Prim and Genetic Algorithms Performance in Determining Optimum Route on Graph*. International Journal of Control and Automation, 109-122. doi:http//dx.doi.org/10.14257/ijca.2018.11.6.11

Valiente Bermudez, F., y Hernandez Velasco, O. (2019). *Ocultamiento del contrato realidad en Rappi Colombia*. Repositorio de la Universidad Simon Bolivar.

Villagómez-Ornelas, P., Hernández-López, P., Carrasco-Enríquez, B., Barrios-Sánchez, K., Pérez-Escamilla, R., & Melgar-Quiñónez, H. (2014). Validez estadística de la Escala Mexicana de Seguridad Alimentaria y la Escala Latinoamericana y Caribeña de Seguridad Alimentaria. salud pública de méxico.

Wollenburg. (2016). Last mile fulfilment and distribution in omni-channel grocery retailing. International Journal of Retail & Distribution Management, Vol. 44. Iss 3 pp. 228 - 247

Zeng, X., Liu, Q., y Yao, S. (2019). *An Improved Prim Algorithm for Connection Scheme of Last Train in Urban Mass Transit Network. Symmetry*, 681-697. doi:https://doi.org/10.3390/sym11050681

Zhou, F., Hu, P., Feng, X., y Song, Y. (2017). *Improved Prim Algorithm and Its Application in Unmanned Aerial Vehicle Cruise System. Kaifeng: Natural Science Fund of China*

ANEXOS

Tabla A

Resultados del Caso 1 (duración)

Algoritmo	Original	Óptimo	Mejora	Mejora %
Kruskal	1742	1721	21	1.21%
Kruskal	2239	2040	199	8.89%
Kruskal	2649	2363	286	10.80%
Kruskal	1678	1466	212	12.63%
Kruskal	2946	2521	425	14.43%
Kruskal	2090	1721	369	17.66%
Kruskal	2102	1714	388	18.46%
Prim	1051	1051	0	0.00%
Prim	2133	2133	0	0.00%
Prim	2585	2585	0	0.00%
Prim	1717	1717	0	0.00%
Prim	2205	2205	0	0.00%
Prim	1051	1051	0	0.00%
Prim	1770	1736	34	1.92%
Prim	1879	1834	45	2.39%
Prim	1626	1562	64	3.94%
Prim	2263	2152	111	4.90%
Prim	2402	2284	118	4.91%
Prim	2316	2202	114	4.92%
Prim	3392	3168	224	6.60%
Prim	1847	1722	125	6.77%
Prim	1852	1714	138	7.45%
Prim	2178	2005	173	7.94%
Prim	1179	1084	95	8.06%
Prim	3145	2873	272	8.65%
Prim	2139	1949	190	8.88%
Prim	1916	1722	194	10.13%
Prim	1588	1409	179	11.27%
Prim	2458	2115	343	13.95%
Prim	2474	2113	361	14.59%
Prim	1189	1015	174	14.63%
Prim	1735	1474	261	15.04%
Prim	1832	1553	279	15.23%
Prim	1823	1537	286	15.69%
Prim	1373	1156	217	15.80%
Prim	1786	1497	289	16.18%
Prim	2029	1678	351	17.30%
Prim	2301	1854	447	19.43%

Tabla B
Resultados del Caso 2 (distancia)

Algoritmo	Original	Óptimo	Mejora	Mejora %
Kruskal	55489	53489	2000	3.74%
Kruskal	29087	27933	1154	4.13%
Kruskal	49388	40381	9007	22.31%
Kruskal	33584	27188	6396	23.53%
Prim	28196	27707	489	1.76%
Prim	27198	26519	679	2.56%
Prim	53287	51714	1573	3.04%
Prim	35085	33431	1654	4.95%
Prim	37139	35074	2065	5.89%
Prim	43909	41081	2828	6.88%
Prim	38082	35458	2624	7.40%
Prim	30523	28225	2298	8.14%
Prim	43511	40166	3345	8.33%
Prim	44638	41194	3444	8.36%
Prim	22410	20305	2105	10.37%
Prim	27212	24446	2766	11.31%
Prim	47897	42785	5112	11.95%
Prim	49219	42811	6408	14.97%
Prim	48599	40866	7733	18.92%
Prim	43457	36091	7366	20.41%
Prim	37044	30160	6884	22.82%
Prim	45788	37026	8762	23.66%

Tabla C
Alojamiento del experimento

Repositorio	github.com/fruizlanders/direcciones-js
Aplicación Node.js	https://direcciones-js.francoruiz2.repl.co/

ALGORITMO MEJORADO BASADO EN KRUSKAL Y PRIM QUE RESUELVE EL PROBLEMA DEL REPARTIDOR ORIENTADO A TIENDAS CON ALMACÉN CENTRALIZADO USANDO DATOS DEL API DIRECTIONS DE GOOGLE

INFORME DE ORIGINALIDAD

10%	7%	3%	6%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad de Lima Trabajo del estudiante	6%
2	riaa.uaem.mx:8080 Fuente de Internet	<1%
3	uvadoc.uva.es Fuente de Internet	<1%
4	kt.ijs.si Fuente de Internet	<1%
5	uptc.edu.co Fuente de Internet	<1%
6	www.slideshare.net Fuente de Internet	<1%
7	hdl.handle.net Fuente de Internet	<1%
8	sexualidad.wordpress.com Fuente de Internet	<1%