

G8



Proyecto de Inteligencia Artificial

Aplicación del Algoritmo de Búsqueda
Ascensión de Colina
Entregable Final

01

Grupo 8

Integrantes:

- Ccolcca Avalos, Mariluz – 19200177
- Estrada Salazar, Jhony Kevin – 15200119
- Noriega Vela, Diego- 19200190
- Ramirez Teran, Brithany Antonella – 19200168
- Sandoval Juarez,Ariana Ayelen – 19200153
- Solis Rivera, Angel Jeanpierre – 19200194

Docente:

- Ing. Gisella Maquen Niño

PRIMERA EXPOSICIÓN

Palabras claves: optimización, búsqueda local, algoritmo de ascenso de colina, empresa de transporte CIVA, camino más corto, aplicación web.

1.

Introducción

Dentro del tema de optimización, sabemos que se busca conseguir la mejor solución posible a cualquier tipo de problema utilizando los recursos disponibles, ya sea maximizar o minimizar. Respecto a esto, para lograr una solución existen muchos algoritmos como el de Ascenso de Colina .

2.

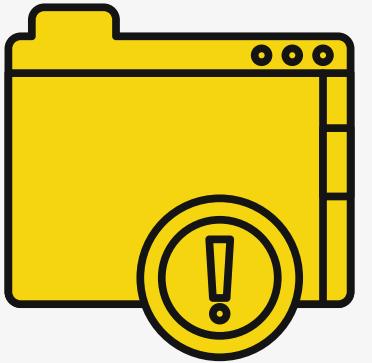
Situación Problemática

Se presenta el capítulo uno con la situación problemática, objetivos y justificación apropiada.

3.

Marco Teórico

El capítulo dos incluye un marco teórico con antecedentes locales, nacionales e internacionales, dentro de las teorías relacionadas tenemos la descripción del algoritmo y herramientas tecnológicas usadas



Ordenar, detallar y corregir
algunos objetivos

G8



Detallar el uso de las
herramientas utilizadas para la
implementación



Modificación del informe final

OBSERVACIONES

Primera Exposición

INTELIGENCIA ARTIFICIAL

04

Objetivos

1. General

- Estudiar el algoritmo de optimización Ascenso de Colina para la resolución del problema búsqueda del camino más corto entre dos puntos cualesquiera de la red de envío de cargos y encomiendas de la empresa de transportes Civa.

2. Específicos

- Desarrollar un código de aplicación web que muestre la implementación del algoritmo Ascenso de colina de manera gráfica.
- Adaptar la lógica del algoritmo de búsqueda Ascenso de Colina dentro del problema objeto de estudio de la empresa.
- Establecer los nodos que serán tomados en cuenta dentro de la codificación del algoritmo tanto en la lógica como en la muestra gráfica.
- Identificar las dificultades y/o errores de la implementación del algoritmo de búsqueda Ascenso de Colina siguiendo la filosofía de esta.
- Implementar y comprobar la eficacia del algoritmo de búsqueda Ascenso de Colina en un determinado sistema.



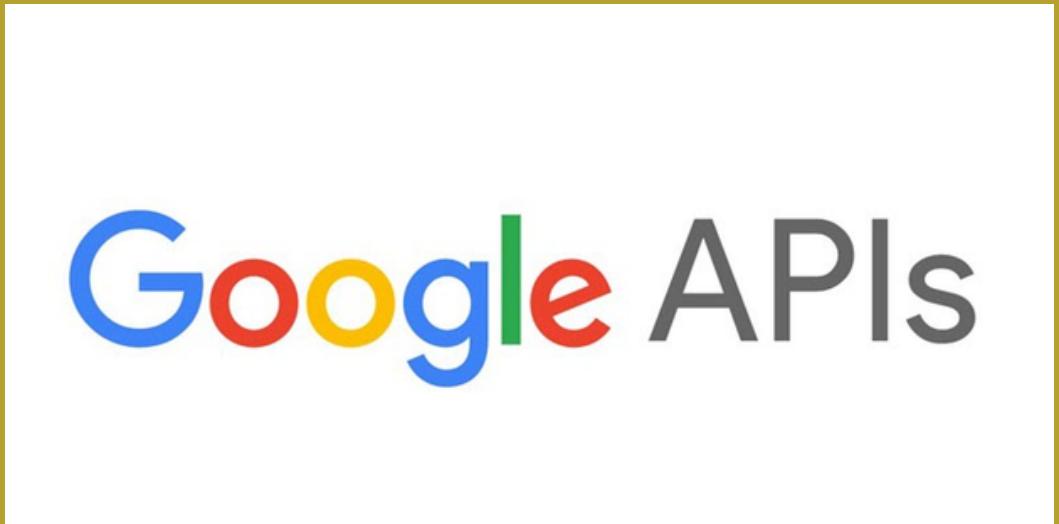
Herramientas tecnológicas

G8

implementación



01



02



JavaScript

03

Herramientas tecnológicas

implementación

G8



03 XAMPP

HTML



CSS



04

05

06

CODIFICACIÓN

1. Solución inicial aleatoria

Siguiendo la lógica del negocio, la solución inicial aleatoria se obtuvo considerando la ubicación de cada una de las agencias disponibles de la empresa

2. Vecinos

Los nodos vecinos pueden afectar al método de búsqueda ya que dentro de la ruta poseen distancias que pueden ser seleccionadas o no por la solución actual

3. Cobertura

En ese sentido lo que se considera es la distancia establecida entre dos nodos como resultado de una fórmula



```
coord={  
    'Sede Javier Prado':[-12.089218091755795, -77.01574868971116],  
    'Sede 28 de Julio':[-12.063226877463787, -77.03352476099145],  
    'Sede Atocongo':[-12.156600691212441, -76.98371335902573],  
    'Sede Terminal Plaza Norte':[-12.005051785087128, -77.05494476760562],  
    'Sede Arequipa Central':[-16.42303396711053, -71.5453360854306],  
    'Sede Arequipa Camaná':[-16.402741432272975, -71.51235965474616],  
    'Sede Piura Loreto':[-5.201091472336525, -80.63118770328701],  
    'Sede Piura Talara':[-4.586560718171606, -81.27041074562112]  
}
```

PASO 1

Establecer los nodos de la red como parte de la información del problema desde la entrada.

PASO 2

Generar una solución inicial aleatoria como 'solución actual'

```
def distancia(coord1,coord2):  
    lat1=coord1[0]  
    lon1=coord1[1]  
    lat2=coord2[0]  
    lon2=coord2[1]  
    c1=((lat1,lon1))  
    c2=((lat2,lon2))  
    return geodesic(c1,c2).km  
  
def i_hill_climbing():  
    #crear ruta inicial aleatoria  
    ruta=[]  
    for ciudad in coord:  
        ruta.append(ciudad)  
    mejor_ruta=ruta[:]  
    max_iteraciones=8  
  
    while max_iteraciones>0:  
        mejora=True  
        #Generamos una nueva ruta aleatoria  
        random.shuffle(ruta)
```

PASO 3

Evaluar la cobertura para todos los vecinos de la solución actual para poder designar mejor vecino.

```
#Calcular distancia cubierta por una ruta
def evalua_ruta(ruta):
    total=0
    for i in range(0,len(ruta)-1):
        ciudad1=ruta[i]
        ciudad2=ruta[i+1]
        total=total+distancia(coord[ciudad1],coord[ciudad2])
        ciudad1=ruta[i+1]
        ciudad2=ruta[0]
        total=total+distancia(coord[ciudad1],coord[ciudad2])
    return total
```

PASO 4

Si la cobertura del mejor vecino es menor que la de la solución actual, se considera mejor solución, se establece como solución actual y vuelve al paso 3. En caso contrario, se arroja la solución actual como la mejor solución del algoritmo en la salida.

```
while max_iteraciones>0:
    mejora=True
    #Generamos una nueva ruta aleatoria
    random.shuffle(ruta)
    while mejora:
        mejora=False
        dist_actual=evalua_ruta(ruta)
        #evaluar vecinos
        for i in range(0,len(ruta)):
            if mejora:
                break
            for j in range(0,len(ruta)):
                if i != j:
                    ruta_tmp=ruta[:]
                    ciudad_tmp=ruta_tmp[i]
                    ruta_tmp[i]=ruta_tmp[j]
                    ruta_tmp[j]=ciudad_tmp
                    dist=evalua_ruta(ruta_tmp)
                    if dist<dist_actual:
                        #Encontrado vecino que mejora el resultado
                        mejora=True
                        ruta=ruta_tmp[:]
                        break
        max_iteraciones=max_iteraciones-1
        if evalua_ruta(ruta)<evalua_ruta(mejor_ruta):
            mejor_ruta=ruta[:]
return mejor_ruta
```

Implementación de la API de Google Maps

PASO 1

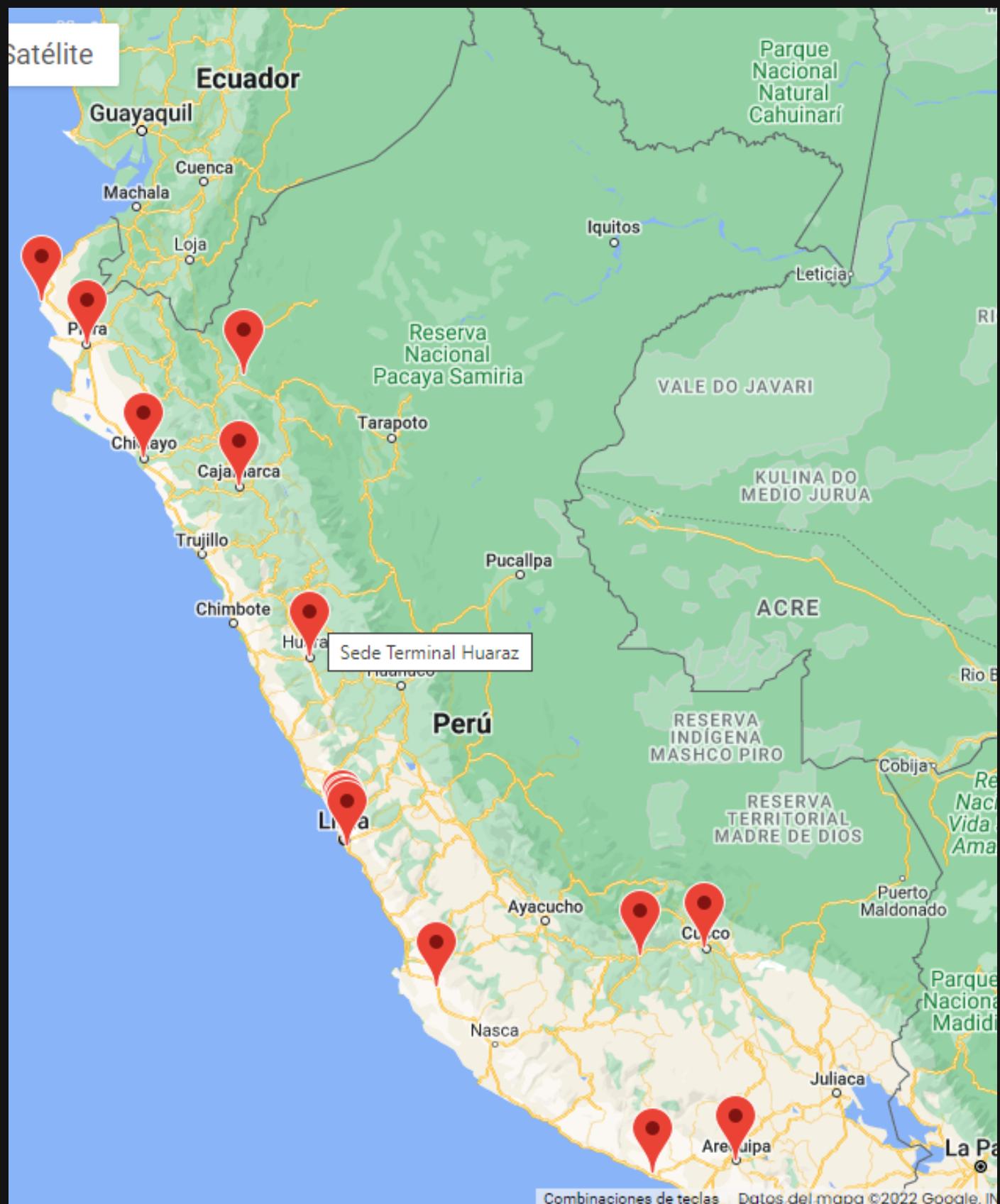
- Instanciar la variable “map”
- Crear la variable “bounds” para guardar las coordenadas de cada punto.
- Crear un arreglo paara guardar la informacion de cada punto.

```
var map=new google.maps.Map(document.getElementById("map"));
var bounds = new google.maps.LatLngBounds();
// Multiple markers location, latitude, and longitude
var markers = [
  ['Sede Javier Prado',-12.089218091755795, -77.01574868971116],
  ['Sede 28 de Julio',-12.063226877463787, -77.03352476099145],
  ['Sede Atocongo',-12.156600691212441, -76.98371335902573],
  ['Sede Terminal Plaza Norte',-12.005051785087128, -77.05494476760562],
  ['Sede Arequipa Central',-16.42303396711053, -71.5453360854306],
  ['Sede Arequipa Camaná',-16.585170973998583, -72.70553736332874],
  ['Sede Piura Loreto',-5.201091472336525, -80.63118770328701],
  ['Sede Piura Talara',-4.586560718171606, -81.27041074562112],
  ['Sede Abancay', -13.639179280953257, -72.88528518972556],
  ['Sede Terminal Bagua Grande', -5.610295414784535, -78.43606746099624],
  ['Sede Terminal Cajamarca', -7.17104578365114, -78.50138495686839],
  ['Sede Terminal Bolognesi Chiclayo', -6.775742750626063, -79.83828664670624],
  ['Sede Terminal Cusco', -13.533136022933917, -71.97171081885351],
  ['Sede Terminal Huaraz', -9.525636997019953, -77.52745021214375],
  ['Sede Terminal Ica', -14.063957987457615, -75.73318988968508]
];
```

PASO 2

Mostrar cada punto en el mapa.

```
// Place each marker on the map
for( i = 0; i < markers.length; i++ ) {
  var position = new google.maps.LatLng(markers[i][1], markers[i][2]);
  bounds.extend(position);
  marker = new google.maps.Marker({
    position: position,
    map: map,
    title: markers[i][0]
  });
  // Center the map to fit all markers on the screen
  map.fitBounds(bounds);
}
```



PASO 3

Trazar las rutas y representarlas en el mapa.

CIVA TRANSPORTES

Agencia de Inicio:

Civa 28 de Julio, PER-LIM

Civa 28 de Julio, PER-LIM
Civa Atocongo, PER-LIM
Civa Terminal Terrestre Plaza Norte, PER-LIM
Civa Javier Prado, PER-LIM
Civa Túnel, PER-LIM

Agencia de Destino:

Civa Arequipa Central, PER-ARE

Enviar



```
const directionsService = new google.maps.DirectionsService();
const directionsRenderer = new google.maps.DirectionsRenderer();
directionsRenderer.setMap(map);

document.getElementById("submit").addEventListener("click", () => {
  calculateAndDisplayRoute(directionsService, directionsRenderer);
});

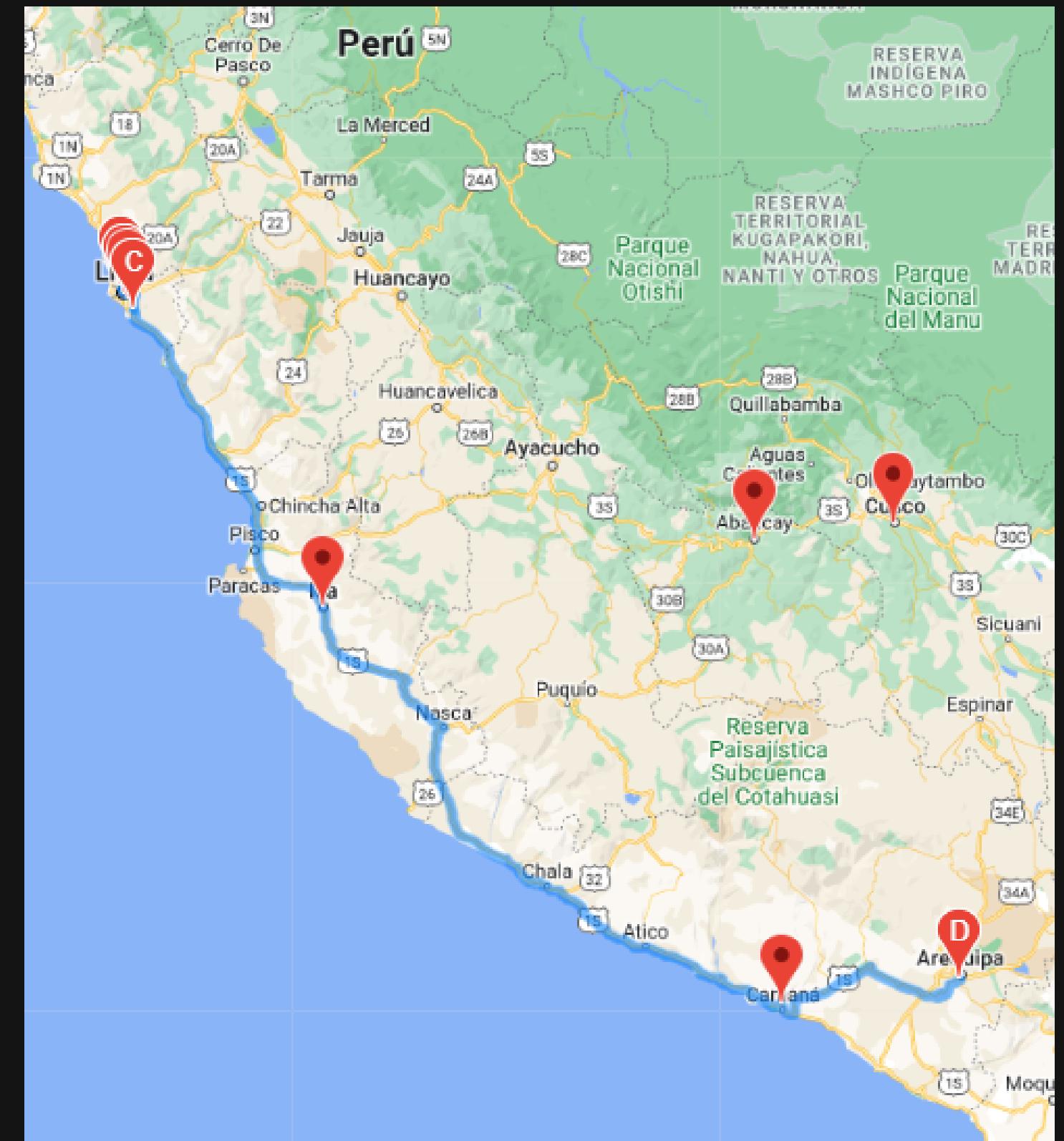
function calculateAndDisplayRoute(directionsService, directionsRenderer) {
  const waypts = [];
  const checkboxArray = document.getElementById("waypoints");

  for (let i = 0; i < checkboxArray.length; i++) {
    if (checkboxArray.options[i].selected) {
      waypts.push({
        location: checkboxArray[i].value,
        stopover: true,
      });
    }
  }
}
```

PASO 4

Se trazar la ruta a través del método "route()" el cual tiene el punto de origen, destino, las terminales intermedias.

```
directionsService
  .route({
    origin: document.getElementById("start").value,
    destination: document.getElementById("end").value,
    waypoints: waypts,
    optimizeWaypoints: true,
    travelMode: google.maps.TravelMode.DRIVING,
  })
  .then((response) => {
    directionsRenderer.setDirections(response);
```



PASO 5

Se muestra la ruta de punto a punto con su respectiva distancia.

```
const route = response.routes[0];
const summaryPanel = document.getElementById("directions-panel");

summaryPanel.innerHTML = "";

// For each route, display summary information.
for (let i = 0; i < route.legs.length; i++) {
    const routeSegment = i + 1;

    summaryPanel.innerHTML +=
`<div class="ruta">
  <h3 class="numeroRuta">Segmento de ruta: ${routeSegment}</h3>
  <div class="ruta-descripcion">
    <span>${route.legs[i].start_address}<b> hacia</b></span>
    <span>${route.legs[i].end_address}</span>
    <p class="distancia">Distancia: ${route.legs[i].distance.text}</p>
  </div>
</div>`
}
```

Segmento de ruta: 1

Av. 28 de Julio 1145, Cercado de Lima 15033, Perú hacia Gran Terminal Terrestre, Cercado de Lima 15311, Perú

Distancia: 9.9 km

Segmento de ruta: 2

Gran Terminal Terrestre, Cercado de Lima 15311, Perú hacia Av. Panamericana sur Km. 11.30, Lima Atocono, Lima 15056, Perú

Distancia: 25.6 km

Segmento de ruta: 3

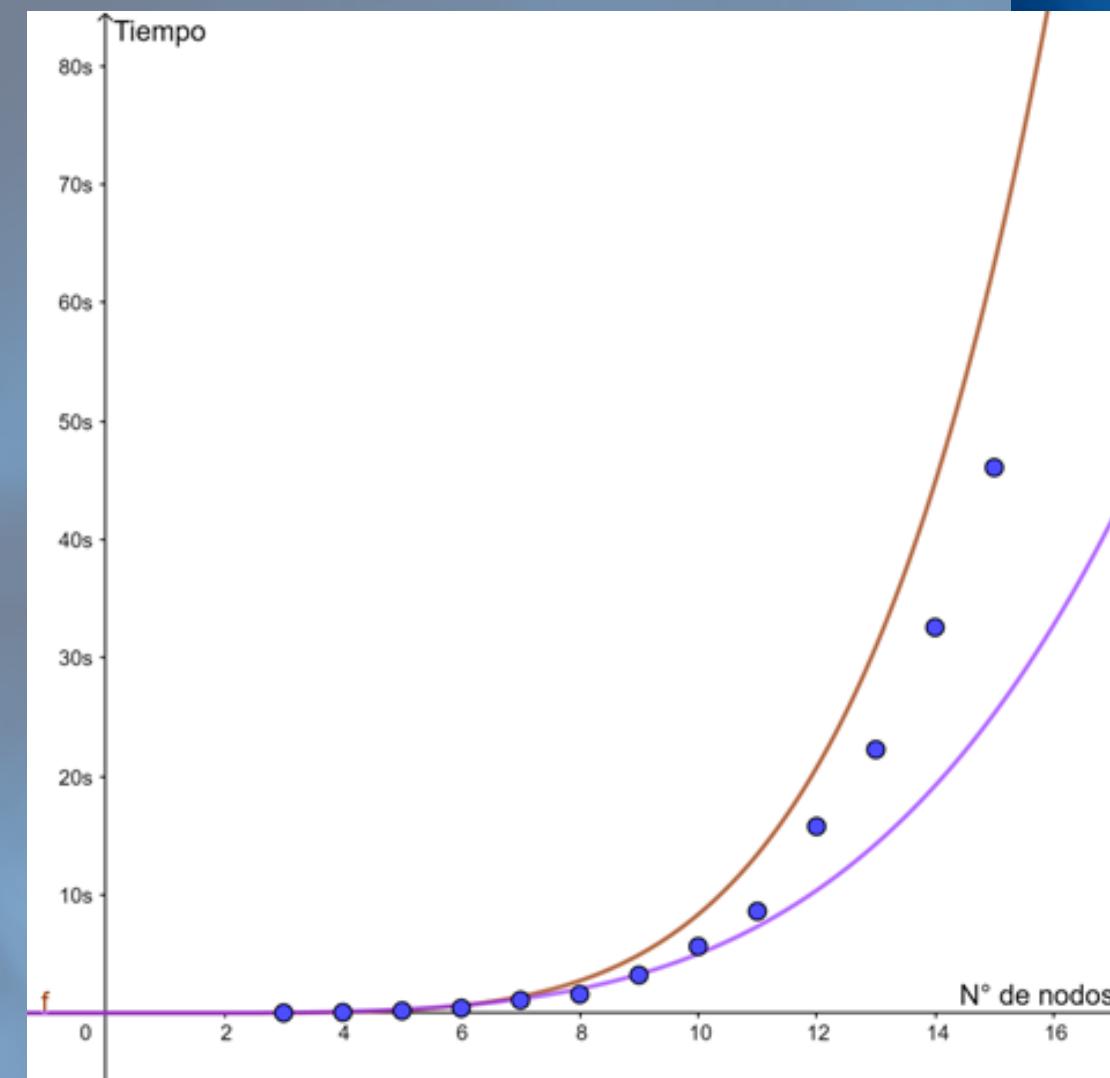
Av. Panamericana sur Km. 11.30, Lima Atocono, Lima 15056, Perú hacia Av. Arturo Ibáñez S/N counter C-41 y C-42, Terrapuerto de Arequipa, Arequipa 04011, Perú

Distancia: 989 km

Evaluación del algoritmo

Resultados obtenidos de medir el tiempo de 99 iteraciones del algoritmo en una misma máquina.

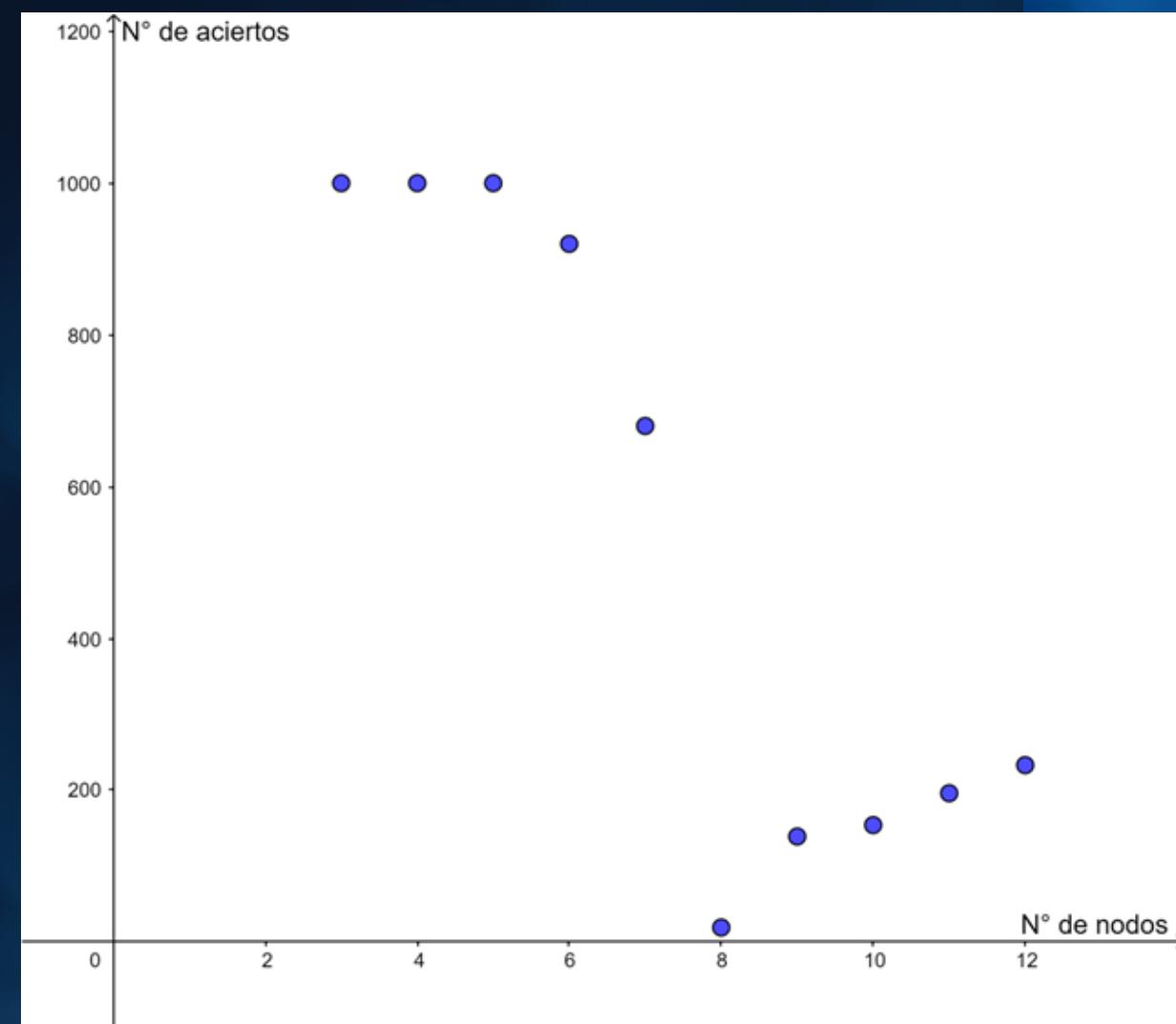
Nº de nodos	Tiempo transcurrido en segundos	Promedio de tiempo por iteración en segundos
3	0.0139122009	0.00014053
4	0.0566909313	0.00057264
5	0.1941697598	0.00196131
6	0.4148068428	0.00418997
7	1.0842208862	0.01095173
8	1.5767636299	0.01592691
9	3.2106478214	0.03243079
10	5.6247177124	0.05681533
11	8.5993726254	0.08686235
12	15.7618877888	0.15921099
13	22.2597634792	0.2248461
14	32.5745806694	0.32903617
15	46.0749831200	0.46540387



Evaluación del algoritmo

Resultados obtenidos de calcular el número de aciertos de 1000 iteraciones del algoritmo

Nº de nodos	Número de aciertos	Porcentaje de aciertos
3	1000	100%
4	1000	100%
5	1000	100%
6	920	92%
7	680	68%
8	19	1.9%
9	139	13.9%
10	154	15.4%
11	196	19.6%
12	233	23.3%

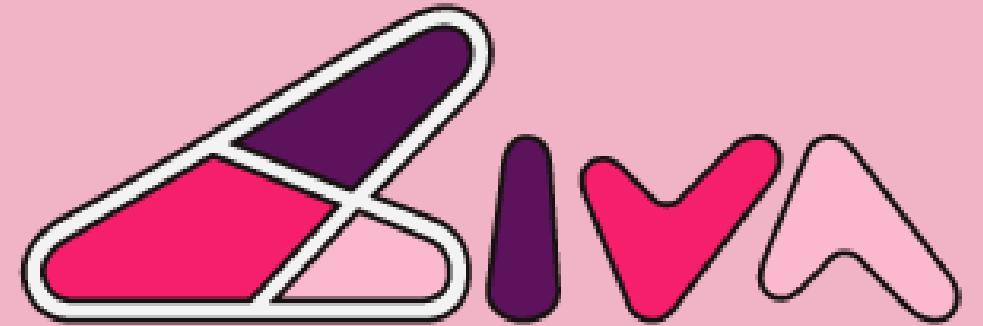


Conclusiones

Según el apartado de antecedentes donde se investiga trabajos con similitud a lo propuesto y conceptos en el marco teórico se logró el correcto estudio del algoritmo “Ascenso de Colina” útil para la resolución del problema de búsqueda del camino más corto entre dos puntos de la red de envío de cargos y encomiendas de la empresa de transportes CIVA.

Gracias a la comprensión del algoritmo de “Ascenso de Colina” se logró adaptar la lógica de este algoritmo dentro del caso propuesto para habilitar la ruta más corta entre dos sedes de la empresa CIVA.

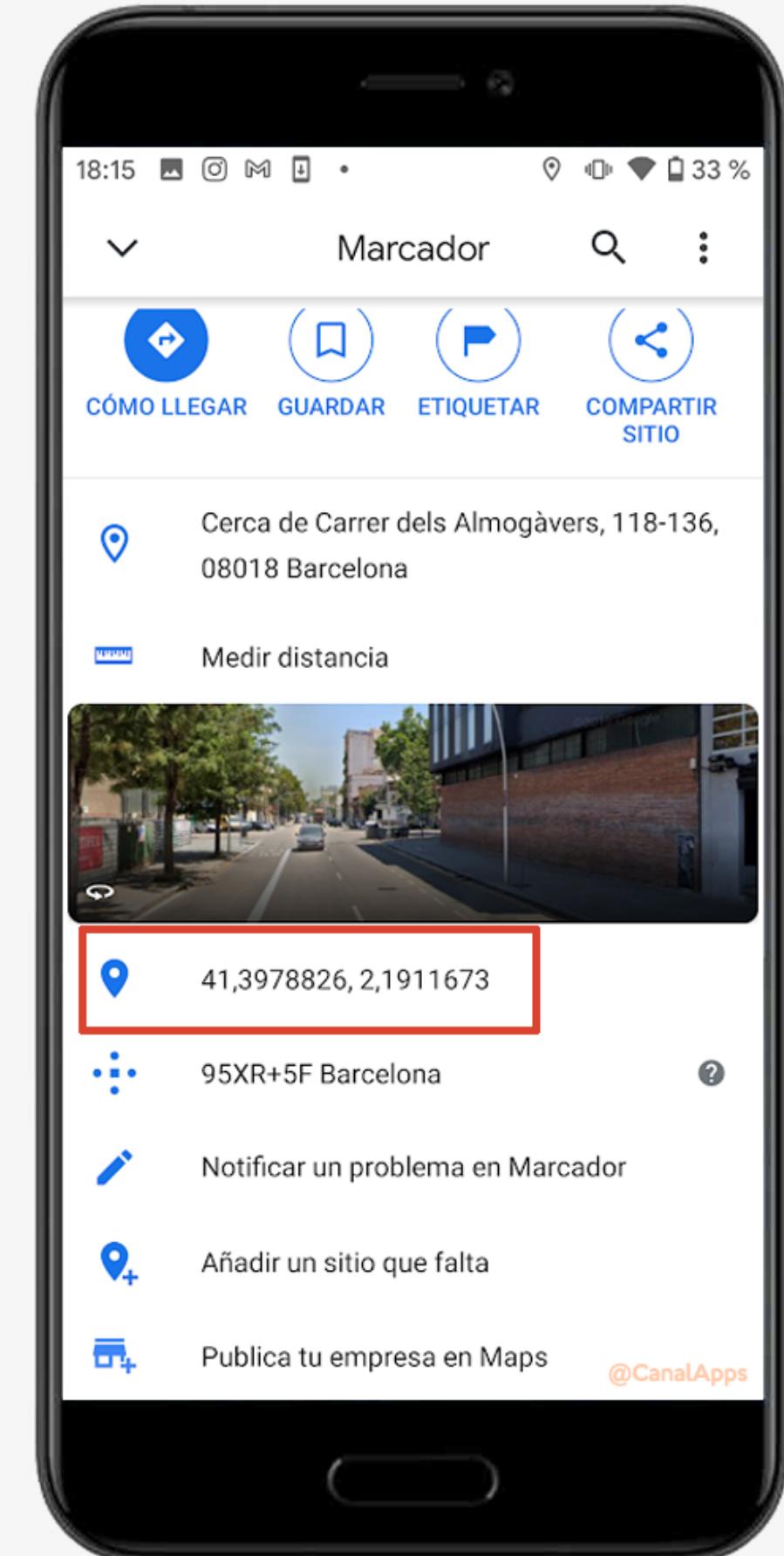
Con el uso correcto de las herramientas tecnológicas propuestas para el caso se logró desarrollar un código de aplicación web de manera local que muestre la implementación del algoritmo “Ascenso de Colina” de manera gráfica.



Dentro del plano de la codificación tenemos que se logra establecer los **nodos considerados las direcciones de las sedes de la empresa de transportes CIVA** con sus respectivas coordenadas que serán tomados en cuenta tanto en la parte lógica como en la muestra gráfica en la aplicación web.

Dentro del plano de las pruebas e implementación se logró identificar las dificultades y/o errores de la adaptación del algoritmo “Ascenso de Colina” para el caso propuesto, específicamente en la **correcta digitación de las coordenadas y nombre de las sedes para la búsqueda de ruta más óptima**.

Respecto al apartado de resultados, tenemos que se logró implementar y comprobar la eficacia del algoritmo de búsqueda “Ascenso de Colina” en un determinado sistema como en este caso.



Recomendaciones

Verificar la correcta instalación de las herramientas tecnológicas.

Dentro de la codificación es importante tomar en cuenta la participación de los nodos, ya que estos son considerados como las sedes de la empresa de transporte CIVA, en ese sentido, se recomienda ser exactos con sus coordenadas y el nombre de la sede dentro de Google Maps.

Respecto a la obtención de la API de Google Maps para la aplicación web, se recomienda utilizar una tarjeta débito solicitada en cualquier entidad bancaria o de algún familiar cercano. Además, investigar sobre la posibilidad de mostrar de manera gráfica con el uso de otra API que sea gratuita y no requiera registro de tarjetas.

Para un mayor análisis se recomienda la implementación de otros algoritmos de búsqueda para diferenciar y saber cual es el más eficiente en este tipo de casos y no solo aplicarlo para empresas de transportes sino también para una escala social como rutas de colegios, avenidas principales, etc.



MUCHAS GRACIAS POR SU ATENCIÓN



INTELIGENCIA ARTIFICIAL

G8

