Informe final problema de Ruteo de Vehículos Eléctricos

Hamilton Tobón Mosquera
Santiago Toro Orozco

Estructuras de datos y Algoritmos
Universidad EAFIT
2018

Tabla de Contenido

1. Introducción

Los vehículos eléctricos son una de las tecnologías más promisorias para reducir la dependencia del petróleo y las emisiones de gases invernadero. El uso de vehículos eléctricos para carga y para el transporte de pasajeros tiene una limitación, y es que el rango de conducción es limitado y el tiempo de carga de la batería es relativamente alto. Por esta razón, es necesario considerar que los vehículos se desvíen de la ruta para ir a estaciones donde puedan recargar su batería [2].
En este informe se describe el problema de rutear vehículos eléctricos, el entendimiento del mismo, una posible solución, que incluye pseudocódigo, complejidad de la solución, además de las ecuaciones usadas para calcular el nivel de batería, la distancia y tiempo y finalmente el código.

2. Problema Ruteo de vehículos de eléctricos

El problema a resolver consiste en **diseñar un algoritmo** para encontrar las rutas óptimas para que un conjunto de carros eléctricos visite un conjunto de clientes. La pregunta es la siguiente:

Dado una lista de clientes ubicados en un mapa vial bidimensional, un depósito de inicio y fin, y unas restricciones de tiempo y energía ¿cuáles son las rutas para un número libre de carros eléctricos, para visitar todos los clientes, minimizando el tiempo total, que es la suma del tiempo del recorrido más el tiempo que toman las recargas de batería?.

3. Entendimiento del problema

Los datos de entrada que me ofrece el problema se refieren al vehículo *peugeot ion* y son los siguientes:

- Coordenadas del punto de salida de los carros o depot, el cual tiene infinitos carros a disposición.
- Coordenadas de las estaciones de recarga de los carros, pues al ser eléctricos tienen un nivel de batería finito.
- Coordenadas de los puntos donde están ubicados los clientes a transportar.
- Tmax: que es el tiempo máximo que puede tardar un carro haciendo un recorrido.
- Smax: que es el tiempo máximo que se tarda una estación recargando un carro.
- Q: Que es la capacidad de la batería de los carros.
- Funciónes de carga de la batería para cada tipo de estación, en términos de *l (tiempo de carga)* y *g(watts)*.

Lo que se debe hacer es recoger a cada cliente, volviendo al punto de salida, pero visitando cada cliente una única vez, recargando el vehículo en medio del viaje si se debe hacerlo y en el menor tiempo posible, teniendo en cuenta que también hay variables como la velocidad del carro, la velocidad de carga de las estaciones, y la tasa de consumo de energía de cada carro. Se puede asumir que hay infinitos carros en el punto de salida.

En resumen hay que encontrar un ciclo hamiltoniano en el grafo dado, en el menor tiempo posible, teniendo en cuenta que el tiempo máximo de ejecución es de 30 segundos.

4. Pseudocódigo de la solución.

a. Macro algoritmo. (Tomado y modificado de [1])

1- Seleccionar punto de salida (depot).
2- Testear la admisibilidad del camino seleccionado hasta ahora.
3- Si el camino seleccionado es VÁLIDO, listar los sucesores del último nodo seleccionado y seleccionar el sucesor más cercano. Repetir paso 1.
4- Si el camino seleccionado es INVÁLIDO, elimine el último nodo seleccionado y seleccione el sucesor siguiente más 5- cercano al antecesor. Repetir paso 1.

6- Si todos los sucesores del actual nodo han sido visitados, y el camino es INVÁLIDO repetir paso 4.
7- Si todos los nodos han sido visitados y el camino es INVÁLIDO entonces no hay ciclo hamiltoniano.
8- Si un sucesor del último nodo visitado es el depot (punto de salida) entonces uno de los ciclos hamiltonianos ha sido encontrado.

El camino seleccionado es VÁLIDO si y sólo si, desde el último nodo agregado puedo volver al nodo de salida, teniendo en cuenta el tiempo máximo de viaje de cada carro, y el nivel de batería en el que está.

b. Pseudocódigo refinado

```
TSP(Grafo g){

  depot = g[0];
  sucesoresDepot = obtenerSucesores(depot)

  caminoActual.add(depot)
  visitados[depot] = 1

  for(i = 0, i < sucesoresDepot.size, i += 1){
    nodoActual = obtenerSucesorMasCercano( sucesoresDepot
)

    if( noVisitado(nodoActual)){

      caminoActual.add( nodoActual )
      if(puedo ir a nodoActual y volver a depot){

          marcarVisitado( nodoActual )
          probarCamino( nodoActual )

        }else
          print "NODO INALCANZABLE"
    }else continue
```

```
      if( esCaminoHamiltoniano )
         print "Ruta " + i + caminoActual

      borrarCaminoActual
    }
  }

  probarCamino( nodoActual ){

    if( ultimo nodo agregado es igual al depot)
      return

    sucesoresNodoActual = obtenerSucesores( nodoActual )

    for(i = 0, i < sucesoresNodoActual.size, i += 1){
      siguienteNodo = obtenerSucesorMasCercano(
  sucesoresNodoActual )

      if( noVisitado( siguienteNodo )){

        caminoActual.add( siguienteNodo )
          if( puedo ir a siguienteNodo y volver a depot ){

             marcarVisitado( siguienteNodo )
             probarCamino( siguienteNodo )

          }else{ //fin camino
            caminoActual.add( depot )
            break
          }
    }
  }
```

c. Complejidad

Complejidad en el peor de los casos de cada función relevante, teniendo en cuenta que *n* equivale al número total de nodos, **no** únicamente clientes. Además las ecuaciones usadas parten

de la ecuación de la recta $Y - Y_1 = m(X - X_1)$ y la ecuación de distancia(x) $x = v * t$:

- obtenerSucesorMasCercano(sucesores) = O(n). Esta función toma en cuenta el nivel de batería y retorna un nodo según la batería actual, y -1 si ya no es posible seguir con otro nodo.
  Para calcular la cantidad de batería usada, se multiplica la tasa de consumo de la batería con el tiempo recorrido hasta el supuesto nodo más cercano.
  - bateriaUsada = tasaDeConsumo * tiempoEnIrAlNodo

  Y el tiempo en ir al nodo se calcula como la distancia que hay entre el nodo actual y este siguiente nodo dividido la velocidad del carro.
  - tiempoEnIrAlNodo = distanciaEuclidianaAlSiguiente / velocidad

- puedoIrAlSiguienteNodoYVolverADepot() = O(n^2). Esta función envuelve dos funciones, prueba si puede ir al siguiente nodo teniendo en cuenta el tiempo máximo de viaje, el tiempo que se tarda en el cliente (si aplica) y el tiempo que se tarda en cada tipo de estación (si aplica).

  Para calcular el tiempo que se tarda en lo diferentes tipos de estaciones, se realizaron los siguientes cálculos:
  - Tasa de carga (m) de cada estación:
    $m = (g_2 - g_1) / (l_2 - l_1)$, teniendo en cuenta que g_2, g_1, l_2, l_1 son los puntos de inicio y fin de la función de carga de la estación.

  - Tiempo (t) que se tarda la estación recargando el carro hasta el tope:
    1. Se calcula la batería que se necesita recargar (gN): gN = nivelBateriaMaximo - nivelBateriaActual
    2. Se calcula tiempo (tAux) que se tarda la estación en recargar el carro hasta gN: tAux = gN / m.

3. El tiempo t entonces es: t = l2 - tAux, puesto que l2 es el tiempo máximo de carga de la estación.

- esCaminoHamiltoniano(caminoActual) = O(1)
- probarCamino(nodoActual) = O(n!)

Por lo tanto la función TSP tiene una complejidad de:

T(n) = n * [O(n) + O(n^2) + O(n!) + O(1)]
T(n) = O(n^2) + O(n) + O(n^3) + O(n*n!) + O(n)
T(n) = **O((n^2) * (n-1)!)**

5. Código

Ubicado en la carpeta electric-vehicle-routing-problem/EVRP/src/evrp. Para compilarlo y ejecutarlo diríjase al fichero *readme.txt* ubicado en electric-vehicle-routing-problem/EVRP.

6. Referencias

[1] A Search procedure for Hamilton Paths and Circuits. Frank Rubin 1973.
[2] https://hal.archives-ouvertes.fr/hal-01245232/document