Práctica 3. Primera parte. Ingeniería de requisitos: Análisis y especificación de requisitos

Noelia Escalera Mejías Alejandro Menor Molinero
Javier Núñez Suárez Adra Sánchez Ruiz
Jesús Torres Sánchez

5 de mayo de 2019

1. Primera heurística

La primera heurística que usaremos es bastante sencilla: escogeremos una ciudad inicial y a partir de ahí seleccionaremos la ciudad más cercana a la última escogida (que no haya sido seleccionada previamente) hasta que no queden ciudades por añadir al circuito. Haremos varias ejecuciones, empezando cada vez de una ciudad distinta, y escogeremos la opción con menos coste.

```
VecinosCercanos(distancias, n, resultado){
  completados;
  todas_las_ciudades;

  // Metemos los indices de las ciudades
  for (i=1; i<=n; i++)
    todas_las_ciudades.insert(i);

  // Iniciamos cada vez en una ciudad diferente
  for(i=1; i<=n; i++){
    candidatos = todas_las_ciudades;
    candidatos.erase(i);
    seleccionados.push_back(i);
    distancia = 0;

    //Creamos el circuito de la ciudad por la que empezamos
    while (!candidatos.empty()){
      actual = seleccionados.back();
      mas_cercano = *candidatos.begin();
      min = distancias[actual][mas_cercano];

      // Averiguamos la ciudad mas cercana
      for(c : candidatos){
        d = distancias[actual][c];
        if (d < min){
          mas_cercano = c;
          min = d;
        }
      }
```

```
30
31          seleccionados.push_back(mas_cercano);
32          distancia += min;
33          candidatos.erase(mas_cercano);
34        }
35        distancia += (distancias[seleccionados.front()][seleccionados
      .back()]);
36
37        completados[distancia] = seleccionados;
38      }
39      resultado = completados.begin()->second;
40    }
41
```

Listing 1: Pseudocódigo de la primera heurística

2. Segunda Heurística

La segunda heurística consiste en conseguir un recorrido parcial que contenga algunas ciudades y posteriormente añadir las ciudades restantes al recorrido. Nosotros hemos buscado la ciudad que esté más al Norte, la que esté más al Sur y la que esté más al Este y a partir de estas tres ciudades hemos formado el resto del circuito.

```
1     Insercion(distancias, n, resultado, ciudades){
2       n = buscarCiudadNorte();
3       s = buscarCiudadSur();
4       e = buscarCiudadEste();
5
6       resultado.aniade(n,s,e);
7       distanciaFinal = distancias[n][e]+distancias[e][s]+distancias
      [s][n];
8
9       for(int i=1;i<=n;i++){
10        if(i!=n && i!=s && i!=e){
11          candidates.insert(i);
12        }
13      }
14
15      while(!candidatos.empty()){
16        for(c:candidatos){
17          for(it=resultado.begin(); it!=resultado.end();it++){
18            siguiente = it;
19            siguiente++;
20
21            if(siguiente == resultado.end())
22              siguiente = resultado.begin();
23
24            diferencia = -distancias[*it][*siguiente];
25            diferencia += distancias[*it][c];
26            diferencia += distancias[c][*siguiente];
27
28             if (it == resultado.begin() || diferencia <
      distanciaMinima){
29            distanciaMinima = diferencia;
```

```
30              insercionMinima = it;
31              }
32            }
33            if (distanciaMinima < calculoMinimo){
34            calculoMinimo = distanciaMinima;
35            posicionMinima = insercionMinima;
36            candidataMinima = c;
37            }
38          }
39          // Borramos de candidatos
40          candidatos.erase(candidataMinima);
41
42          // Insertamos
43          posicionMinima++;
44          resultado.insert(posicionMinima, candidataMinima);
45
46          // Actualizamos la distancia con la insercion
47          distanciaFinal += calculoMinimo;
48        }
49        return distanciaFinal;
50      }
51
```

Listing 2: Pseudocódigo de la segunda heurística