RCVRP - Estado de avance

...

Felipe Rojas
201873112-9

Idea detrás del código

- Representación
- Greedy
- Función de evaluación
- Greedy como solución inicial
- HCAM-Restart

Representación

La representación escogida es la de vector, en cada índice del vector se encontrarán los nodos visitados.

0	5	3	2	1	4	6

Esta representación tiene la ventaja de poder mostrar el orden de visita de cada nodo

Greedy

El algoritmo Greedy o Voraz nos ayudará a crear una solución que utilizaremos más adelante para la siguiente implementación.

Para este caso, el algoritmo Greedy utilizado se basa en encontrar el nodo más cercano al nodo actual, partiendo siempre desde el origen.

```cpp
int mas_cercano(Punto origen, vector<Punto> nodos, vector<bool> visitados) {
    float minimo = INFINITY;
    int mas_cercano = -1;

    for (int i = 0; i < nodos.size(); i++) {
        if (!visitados[i]) {
            float d = distancia(origen, nodos[i]);

            if (d < minimo) {
            mas_cercano = i;
            minimo = d;
            }
        }
        //cout << "nodo: " << i << " distancia: " << d << endl;
    }

    return mas_cercano;
}
```

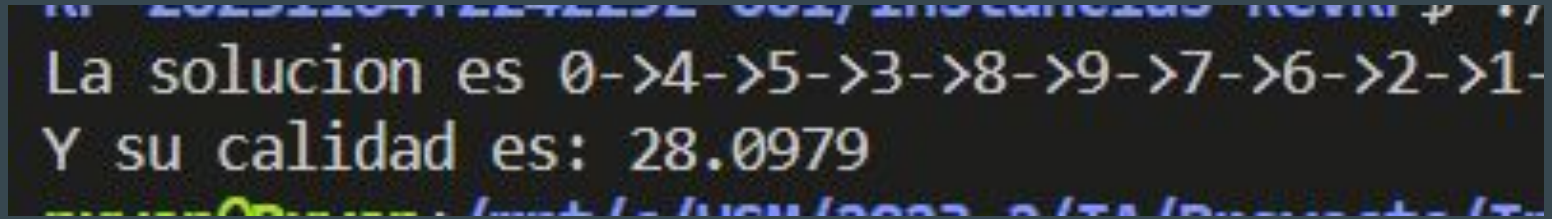
```cpp
vector<int> greedy(Punto origen, vector<Punto> nodos, vector<bool> visitados) {
    vector<int> ruta(visitados.size() + 1);

    ruta[0] = 0;
    visitados[0] = true;

    int indice = mas_cercano(origen, nodos, visitados);

    origen = nodos[indice];
    for (int i = 1; i < visitados.size(); i++) {
        indice = mas_cercano(origen, nodos, visitados);
        ruta[i] = indice;
        visitados[indice] = true;
        origen = nodos[indice];
    }

    return ruta;
}
```



Instancia SET O “10.txt”

Función de evaluación

La función de evaluación se encarga de mostrarnos la calidad de la solución, esta también actúa de cierta forma como función miope para el algoritmo Greedy explicado anteriormente.

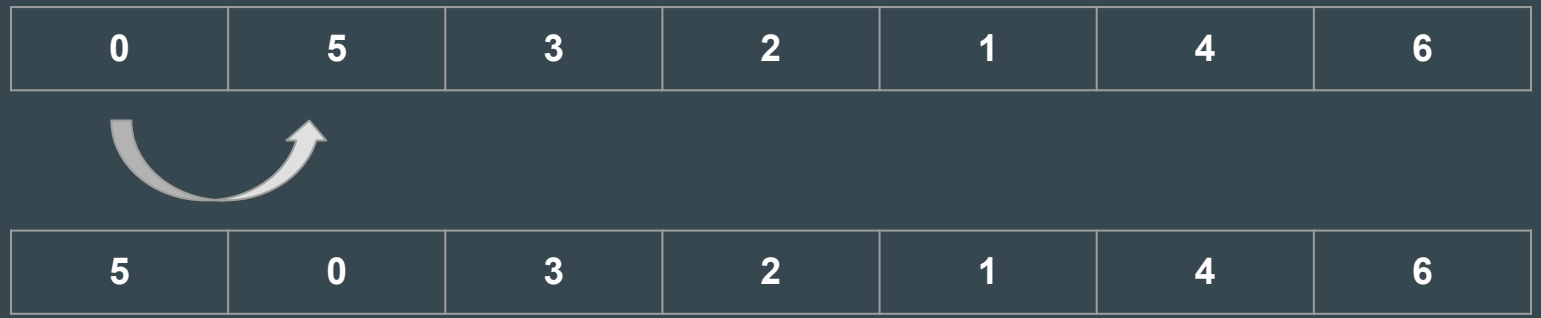
```
float funcion_objetivo(vector<int> vector_solucion, vector<Punto> nodos) {
    float value = 0;
    for (int i = 0; i < vector_solucion.size() - 1; i++) {
        value = value + distancia(nodos[vector_solucion[i]], nodos[vector_solucion[i + 1]]);
    }
    return value;
}
```

La función, como se muestra en el modelo matemático, sólo comprende la distancia recorrida.

Greedy como solución inicial

El algoritmo Greedy presentado anteriormente, nos entrega una solución al RCVRP. Utilizando la misma representación, y esta solución como un punto de partida, aplicaremos movimientos sobre ella a modo de exploración, además de un algoritmo HCAM para encontrar una solución de mayor calidad.

El movimiento utilizado sobre la solución obtenida por Greedy será Swap, este movimiento se aplicará entre nodos visitados consecutivamente.



HCAM - Restart

Aplicando el movimiento para generar una vecindad sobre la solución inicial, se realizará la primera iteración del algoritmo HCAM. Se calculará la calidad de cada solución generada por el vecindario con la función objetivo mostrada anteriormente, cuando exista una mejora, esta será la nueva solución y se realizará una nueva iteración.

En caso de que el algoritmo se atasque en una solución, se reiniciará con una nueva solución.