UNIVERSIDAD SIMÓN BOLÍVAR
Dpto. de Computación y Tecnología de la Información
CI-2693 - Lab. de Algoritmos y Estructuras de Datos III

Proyecto II

Un algoritmo heurístico para resolver el Problema del Cartero Rural

Autor:
Christopher Gómez (18-10892)
Ka Fung (18-10492)

Profesor:
Guillermo Palma

16 de enero de 2022

1. Introducción

El siguiente estudio experimental consiste en implementar un algoritmo heurístico para obtener soluciones aproximadas al Problema del Cartero Rural (RPP), modelado mediante el uso de grafos no dirigidos. Para ello, se empleará el algoritmo presentado por Pearn y Wu, el cual es una modificación del algoritmo de Christofides et al. El RPP consiste en encontrar un ciclo de costo mínimo en un grafo no dirigido, tal que atraviese un conjunto de lados requeridos al menos una vez.

En el algoritmo de Pearn y Wu requiere de la implementación de un algoritmo de apareamiento perfecto de costo mínimo. Por ello, se implementará dos algoritmos heurísticos para obtener un apareamiento perfecto de un grafo completo (un algoritmo ávido y el Vertex-Scan presentado por David Avis [1]), que proporcionan una solución aproximada y que sirven como una alternativa mucho más eficiente y menos compleja que el algoritmo de Edmonds, el cual proporciona una solución exacta al problema.

Luego, el objetivo de este estudio consiste en comparar la eficacia y eficiencia de estos dos algoritmos aplicados a solucionar 36 distintas del problema RPP.

2. Diseño de la solución

Construcción de grafo GR y G' (mapeo G -¿GR)

Construcción del grafo G0 (mapeo GR -¿G0)

Implementación del algoritmo de Dijkstra para grafos no dirigidos

Implementación del ciclo euleriano para grafos no dirigidos

3. Detalles de la implementacion

Algoritmo para obtener las componentes conexas en grafos no dirigidos.

Algoritmo de Dijkstra para obtener los caminos de costo mínimo en grafos no dirigidos.

Algoritmo de Prim para obtener el árbol mínimo cobertor.

Algoritmo para obtener el ciclo euleriano de grafos no dirigidos.

Algoritmo de apareamiento perfecto ávido.

Algoritmo de apareamiento perfecto Vertex-Scan.

Programa cliente para la solución del problema RPP.

4. Detalles de la plataforma

Los siguientes son los detalles de la máquina y el entorno donde se ejecutaron los algoritmos:

- **Sistema operativo**: GNU/Linux (Linux Mint 20 Ulyana).
- **Procesador**: Intel(R) Core(TM) i7-8750H.
- **Memoria RAM**: 8GB.
- **Compilador**: Kotlin version 1.5.31 (JRE 11.0.9.1+1-Ubuntu-0ubuntu1.20.04).
- **Entorno de ejecución**: OpenJDK Runtime Environment (11.0.9.1).

5. Resultados experimentales

Para realizar una comparación entre la solución obtenida y la solución óptima de una instancias del problema de RPP, se calculó el porcentaje de desviación con la fórmula:

$$\%desv = \frac{\text{valor obtenido} - \text{valor óptimo}}{\text{valor óptimo}} * 100 \tag{1}$$

hola otra vez, aquí van dos tablas.

La primera tabla debe tener los valores obtenidos en la solución de las instancias de RPP. En especifico la tabla debe mostrar:
1. El nombre de la instancia,
2. El valor óptimo de la instancia,
3. El porcentaje de desviación de la solución obtenida usando la heurística ávida del Algoritmo 2.
4. El porcentaje de desviación de la solución obtenida usando la heurística Vertex-Scan del Algoritmo 3

Para el caso de la heurística Vertex-Scan, para cada instancia debe presentar el promedio de las soluciones de tres corridas. La segunda tabla debe mostrar el tiempo promedio que tomó la ejecución de todas las instancias, usando heurística ávida y la heurística Vertex-Scan.

6. Análisis de los resultados

7. Conclusiones

-
-

8. Referencias (en caso de tenerlas)

ñema.

Nombre de la instancia	Valor óptimo	Desviación (%)	
		Alg. Ávido	Vertex-Scan
UR132	**23913**	16,271	20,655
UR135	**33088**	14,335	16,814
UR137	**42797**	10,347	13,594
UR142	**25548**	15,935	19,035
UR145	**39008**	7,342	12,603
UR147	**55959**	6,346	7,604
UR152	**28975**	11,365	17,336
UR155	**49156**	4,669	7,078
UR157	**70231**	3,91	4,595
UR162	**32341**	10,998	12,462
UR165	**58800**	4,978	6,2
UR167	**82481**	2,917	3,572
UR532	**17277**	16,438	22,261
UR535	**23635**	16,306	16,839
UR537	**30098**	9,253	12,324
UR542	**17830**	14,173	20,269
UR545	**29648**	6,365	8,835
UR547	**38692**	4,784	7,877
UR552	**20097**	12,579	16,875
UR555	**34488**	4,126	8,069
UR557	**48307**	4,651	3,97
UR562	**24556**	7,188	10,541
UR565	**42828**	4,056	5,17
UR567	**58971**	4,087	4,015
UR732	**21114**	18,751	21,313
UR735	**28663**	10,623	15,26
UR737	**36588**	6,308	11,45
UR742	**22557**	13,078	17,916
UR745	**32493**	10,19	10,96
UR747	**47764**	5,481	6,385
UR752	**25131**	12,614	15,796
UR755	**41774**	4,517	8,357
UR757	**58416**	4,002	5,876
UR762	**27880**	8,691	13,155
UR765	**50492**	4,565	5,696
UR767	**72950**	3,413	3,911

Cuadro 1: Desviación de los resultados obtenidos para cada algoritmo