Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-2693 - Laboratorio de Algoritmos y Estructuras III
Trimestre Septiembre-Diciembre 2021

Proyecto 2: Un algoritmo heurístico para resolver el Problema del Cartero Rural

1. Sobre el Problema del Cartero Rural

El objetivo de este proyecto es la implementación de un algoritmo heurístico constructivo para resolver el Problema del Cartero Rural, en inglés *The Rural Postman Problem* (RPP). El RPP es problema clásico de optimización de enrutamiento por arcos.

EL RPP es un problema NP-*Hard* [6]. Existen problemas de la vida real que pueden ser modelados como RPP. Eiselt et al. [5] indica que entre las aplicaciones prácticas de los problemas de enrutamiento por arcos encontramos: la planificación del mantenimiento de calles, la entrega de correo, la recolección de basura, el despeje de caminos con nieve y la inspección de lineas eléctricas, entre otros.

El objetivo principal de este proyecto es la implementación de un algoritmo heurístico que obtenga un soluciones aproximadas del RPP. En específico se requiere que implemente el algoritmo constructivo presentado por Pearn y Wu [8], el cual es una modificación del algoritmo propuesto por Christofides et al. [3]. Benavent et al. [2] demostraron que con el algoritmo de Christofides en el peor caso se obtienen soluciones que son 3/2 del valor óptimo. Es decir, siempre se cumple que $\frac{\text{Valor Christofides et al.}}{\text{Valor Óptimo}} \leqslant 3/2$.

2. Preliminares

Antes de mostrar el algoritmo para resolver el RPP, se presentan unas definiciones preliminares que son necesarias para la realización del proyecto.

Definición 1 (Problema del Cartero Rural)**.** *Dado un grafo grafo no dirigido y conectado $G = (V, E)$, y dado un conjunto de lados requeridos R, tal que $R \subseteq E \wedge R \neq \emptyset$, en donde los lados $(i, j) \in E$ tienen un costo $c_{ij} \geqslant 0$. Se quiere encontrar un ciclo en E de costo mínimo, tal que atraviesa todos los lados en R al menos una vez.*

Definición 2 (Grafo Par)**.** *Grafo cuyo todos sus vértices tienen grado par.*

Definición 3 (Apareamiento perfecto de costo mínimo)**.** *Sea $G = (V, E)$ un grafo no dirigido, completo, con un número par de nodos $|V| = n$ y con función de costo no negativa $w = E \to \mathbb{R}^+$ sobre el conjunto de lados. Un apareamiento perfecto del conjunto V consiste en un subconjunto de lados $M \subseteq E$, tal que ninguno de los lados de M son adyacentes, y cada vértice de V es incidente con exactamente un lado de M. El costo $w(M)$ del apareamiento M es la suma de los costos de sus lados. Se quiere obtener apareamiento M de costo mínimo.*

Una medida que es útil cuando se reporta resultados de algoritmos aproximados, es el porcentaje de desviación de la solución obtenida con respecto a la solución óptima. A esta medida la denotamos como $\%desv$ y se calcula como sigue:

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

3. Descripción del algoritmo heurístico para el RPP

El Algoritmo 1 presenta la modificación del algoritmo de Christofides para el RPP propuesta por Pearn y Wu [8].

4. Algoritmos heurísticos para el apareamiento perfecto

La solución apareamiento perfecto de costo mínimo, puede ser obtenida con el algoritmo de Edmonds [4]. Kolmogorov [7] presentó una implementación eficiente de éste algoritmo. Debido a la complejidad en la implementación de los algoritmos que obtienen el apareamiento perfecto de costo mínimo, se van a implementar dos algoritmos heurísticos para obtener un apareamiento perfecto de un grafo completo, que nos proporcionan una solución aproximada. Uno de los objetivos de este proyecto va a ser el de comparar la efectividad de los dos algoritmos aproximados de apareamiento perfecto, usando como grafos de pruebas, los grafos completos que se obtienen durante la solución del RPP. A continuación se presentan los dos algoritmos heurísticos que debe implementar, el Algoritmo 2 es un algoritmo ávido (*greedy*), mientras que el Algoritmo 3 fue presentado por David Avis [1].

5. Ejemplo ilustrado

A continuación se presenta un ejemplo que ilustra la aplicación del Algoritmo 1, usando para obtener el apareamiento perfecto el Algoritmo 2. Sea el grafo de la Figura 1a la instancia RPP la cual se quiere solucionar. Los lados requeridos R, generan el grafo inducido G_R que se muestra en la Figura 1b, el cual tiene 3 componentes conexas. Luego se construye un grafo completo en donde cada nodo representa a una componente conexa. El grafo resultante se muestra en la Figura 1c, donde el costo entre cada vértice corresponde al valor del camino de costo mínimo en el grafo original de la Figura 1a, entre los vértices de las componentes conexas de la Figura 1b. El siguiente paso es computar el árbol mínimo cobertor del grafo de la Figura 1c, obteniéndose el árbol de la Figura 1d. Los lados asociados al árbol mínimo cobertor, esto es el conjunto $E_t = \{(2,4),(2,7)\}$, se agregan al grafo G' el resultado se puede observar en la Figura 1e. El siguiente paso es determinar los vértices de grado impar V_o de G' y crear un grafo completo con ellos. Se tiene que $V_o = \{1,3,5,6\}$ y los costos de los lados se obtienen de los valores de los caminos de costo mínimo del grafo original de la Figura 1a. El grafo resultante se muestra en la Figura 1f. De éste grafo se obtiene el apareamiento perfecto de costo mínimo que da como resultado los lados del conjunto $M = \{(1,3),(5,6)\}$. Los lados del conjunto M están asociados a caminos de costo mínimo en el grafo original que contienen un sólo lado, por lo que se agrega al grafo G' el lado $(1,3)$ y el lado $(5,6)$, el resultado se observa en la Figura 1g. El grafo de la Figura 1g es un grafo par, por lo que se puede obtener un ciclo euleriano que es la solución del RPP. El ciclo que se obtiene es $1-2-4-5-6-7-2-3-1$ y el costo de la solución es 30. La solución óptima tiene un costo de 26 y se muestra en la Figura 1h.

Algoritmo 1: Algoritmo heurístico para obtener una solución factible para el RPP

Entrada: Un grafo no dirigido y conexo $G = (V, E)$, y un subconjunto $R \subseteq E$ de lados requeridos

Salida: Un ciclo $\mathscr{C}$ de G que es solución factible de RPP

1 **inicio**
2 Crear un grafo $G_R = (V_R, R)$, en donde V_R son los vértices de los lados de R;
3 $G' \longleftarrow G_R$;
4 **si** G' *es conexo* **entonces**
5 **si** G' *es par* **entonces**
6 ir a línea 24;
7 **de lo contrario**
8 ir a línea 16;
9 Sean $\{C_1, C_2, \ldots, C_n\}$ el conjunto de **componentes conexas** de G', Se construye un grafo completo $G_t = (V_t, E_t)$, donde cada $v_i \in V_t$ corresponde a una componente conexa C_i, por lo que $|V_t| = n$. Cada lado $e_t \in E_t$ tiene un costo dado por la función $c_{e_t} : E_t \to \mathbb{R}^+$ que se define como:
10 $$c_{e_t}(e_t) = \text{mín}\{spl(v_i, v_j) \mid v_i \in C_i \wedge v_j \in C_j \wedge i \neq j\}$$
11 donde *spl* es una función que retorna el valor del **camino de costo mínimo** entre los vértices v_i y v_j, al cual llamamos CCM_{v_i,v_j}, en el grafo G . Cada lado $e \in E_t$ tiene asociado un CCM_{v_i,v_j};
12 Obtener el **árbol mínimo cobertor** del grafo G_t, donde E_{MST} son el conjunto de lados del árbol. Sean E_{t0} el conjunto de lados que corresponden a los lados de los CCM_{v_i,v_j} asociados a los lados de E_{MST};
13 Se simplifica E_{t0} eliminando todos los lados duplicados. Se denomina a este nuevo conjunto de lados como E_t ;
14 Se agregan a G' vértices en E_t que no se encuentren V_R ;
15 Se agrega a G' los lados E_t, se permite que hayan lados duplicados;
16 Determine el conjunto de vértices V_0 de grado impar en G'. Se construye un grafo completo G_0 teniendo como vértices a V_0. Cada lado (v_i, v_j) de G_0 tiene como costo el valor del camino de costo mínimo CCM_{v_i,v_j} entre los vértices v_i y v_j en G. Observe que cada lado (v_i, v_j) tiene asociado un CCM_{v_i,v_j} ;
17 Se determina el **apareamiento perfecto de costo mínimo** de G_0, al conjunto de lados del apareamiento se le denomina como M ;
18 **ParaCada** *lado* $(v_i, v_j) \in M$ **hacer**
19 Obtener el CCM_{v_i,v_j} asociado a (v_i, v_j);
20 **ParaCada** *lado* $(i, j) \in CCM_{v_i,v_j}$ **hacer**
21 **si** $i \notin G'$ **entonces** se agrega el vértice i a G';
22 **si** $j \notin G'$ **entonces** se agrega el vértice j a G';
23 Se agrega el lado (i, j) a G' sin importar que se encuentre duplicado;
24 En este punto G' debe ser par, entonces obtener el **ciclo euleriano** $\mathscr{C}$ de G' ;
25 **retorna** $\mathscr{C}$;

6. Detalles de la implementación

Se puede observar que varios de los algoritmos implementados en `grafoLib`, son requeridos en para la implementación del algoritmo para el RPP. Entre ellos se encuentran: (I) el

Algoritmo 2: Algoritmo ávido para obtener un apareamiento perfecto

Entrada: Un grafo $G = (V, E)$ no dirigido, completo y con un número par de vértices

Salida: Un conjunto de lados $\mathscr{M}$ que es un apareamiento perfecto de G

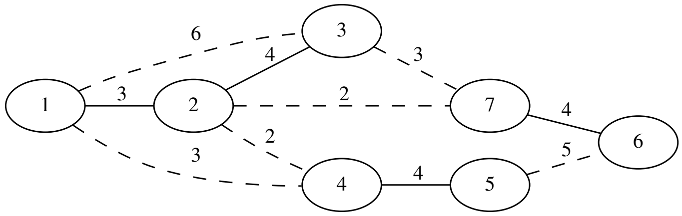
inicio
    $\mathscr{M} \longleftarrow \emptyset$;
    $V' \longleftarrow V$;
    Construir una lista de lados L que contiene los lados de E ordenados en forma ascendente a su costo ;
    mientras $V' \neq \emptyset$ **hacer**
        $(i, j) \longleftarrow$ desencolar el tope de L ;
        si $i \in V' \wedge j \in V'$ **entonces**
            Agregar el lado (i, j) a $\mathscr{M}$;
            Eliminar los vértices i y j de V';
    retorna $\mathscr{M}$;

Algoritmo 3: Algoritmo Vertex-Scan para obtener un apareamiento perfecto

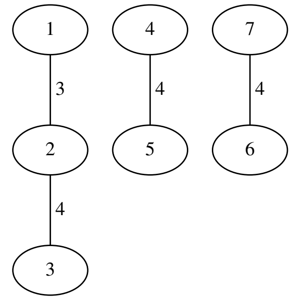
Entrada: Un grafo $G = (V, E)$ no dirigido, completo y con un número par de vértices

Salida: Un conjunto de lados $\mathscr{M}$ que es un apareamiento perfecto de G

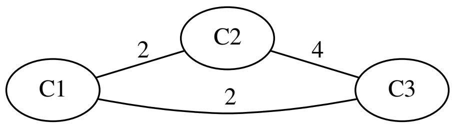
inicio
    $V' \longleftarrow V$;
    $E' \longleftarrow E$;
    mientras $V' \neq \emptyset$ **hacer**
        Escoger un vértice $i \in V'$ aleatoriamente ;
        Escoger el lado $(i, j) \in E'$ con menor costo ;
        Agregar (i, j) a $\mathscr{M}$;
        Eliminar los vértices i y j de V';
        Eliminar de E' todos los lados que tenga como adyacentes los vértices i o j;
    retorna $\mathscr{M}$;



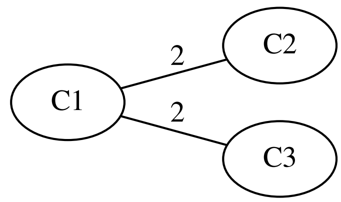
(a) Grafo instancia de RPP, los lados sólidos son los lados requeridos R.



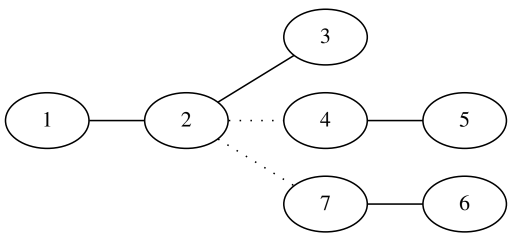
(b) Grafo G_R



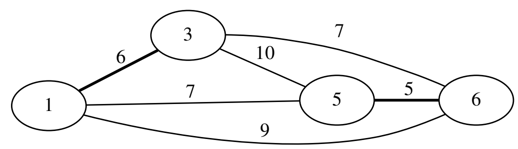
(c) Grafo G_{cc} donde los vértices son las componentes conexas de G_R.



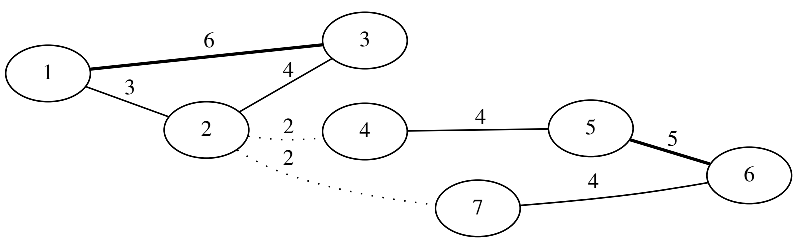
(d) Árbol mínimo cobertor del grafo grafo G_{cc}.



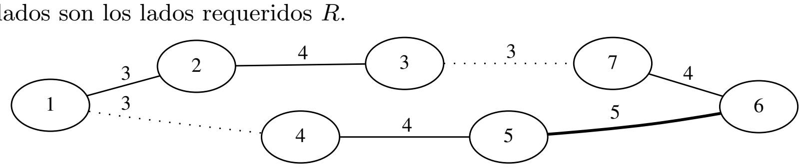
(e) Grafo G' resultado de G_R más los lados E_t, los lados E_t en líneas con puntos.



(f) Grafo completo a para determinar el apareamiento perfecto de costo mínimo, los lados gruesos son la solución del apareamiento perfecto de costo mínimo.



(g) Grafo G' par que contiene la solución del RPP. Los lados gruesos son los que provienen en el apareamiento perfecto de costo mínimo y los lados con puntos provienen de la solución del árbol mínimo cobertor. Los restantes lados son los lados requeridos R.



(h) Grafo que es la solución óptima de la instancia RPP.

Figura 1: Grafos que se producen al resolver una instancia del RPP.

algoritmo de detección de componentes conexas, (II) el algoritmo de costo mínimo entre un par de vértices de un grafo, (III) el algoritmo de obtención del árbol mínimo cobertor, y (IV) el algoritmo de obtención de un ciclo euleriano. La implementación de los algoritmos de este proyecto deben hacer uso de `grafoLib`. Para ello debe realizar las modificaciones que sean necesarias a `grafoLib`. Entre ellas están, la que permita tener lados múltiples en los grafos no dirigidos. También es necesario implementar una versión del algoritmo de Dijkstra que trabaje con grafos no dirigidos. La implementación de esta versión debe hacerse, como parte de la librería `grafoLib`, en una clase llamada `DijkstraGrafoNoDirigido`, en un archivo llamado `DijkstraGrafoNoDirigido.kt`. La firma del constructor y de los métodos deben ser análogas a las presentadas en `Dijkstra.kt`.

El Algoritmo 2 debe implementarse en una clase llamada `ApareamientoPerfectoAvido`, en un archivo llamado `ApareamientoPerfectoAvido.kt`. El Algoritmo 3 debe estar contenido en una clase llamada `ApareamientoVertexScan`, en un archivo llamado `ApareamientoVertexScan.kt`. Estos dos algoritmos de aparemiento perfecto, deben ser parte de `grafoLib`.

El Algoritmo 1 debe implementarse como un programa cliente que hace `grafoLib`, en un archivo llamado `HeuristicaRPP.kt`. Para se ejecución debe crear un programa llamado `runHeurRPP.sh`. El comando para su ejecución es como sigue:

```
>./runHeurRPP.sh [a|v] <instancia>
```

donde las opciones `a` y `v` corresponden a la utilización del Algoritmo 2 y del Algoritmo 3 respectivamente. El parámetro `<instancia>` corresponde al nombre del archivo con la instancia RPP a resolver. La llamada a `runHeurRPP.sh` debe hacerse con exactamente dos parámetros, es decir, con una opción de heurística de apareamiento y con la instancia a resolver. En otro caso el programa indica que hubo un error con los parámetros de entrada.

La salida del programa `runHeurRPP.sh` consiste de 3 líneas. La primera línea contiene el ciclo que es solución del problema RPP mostrando los vértices del ciclo separados por un espacio en blanco. La segunda línea muestra el costo de la solución obtenida. La tercera y última línea indica el tiempo, en segundos, que tomó al programa obtener la solución del problema, contando la carga de la instancia. Por ejemplo, dada la instancia del ejemplo de la Figura1, una salida válida es como sigue:

```
1 2 4 5 6 7 2 3 1
30
0.134 segs.
```

Además de los códigos fuentes en `Kotlin`, debe entregar un archivo `Makefile` que compile todo el proyecto. El código debe estar debidamente documentado y debe seguir la guía de estilo dada en clase. La implementación de los algoritmos deben ser *razonablemente eficiente*. Se deben usar estructuras de datos que sean apropiadas para cada problema. Estos aspectos serán tomados en cuenta en la evaluación.

Si un proyecto que *no pueda compilarse o ejecutarse tiene cero de nota*. La plataforma en la que debe compilarse y ejecutarse el proyecto es GNU/Linux.

7. Estudio experimental

Se desea hacer un estudio experimental en donde se comparan los resultados de resolver instancias RPP, por medio de las dos heurísticas de apareamiento perfecto. Las instancias del RPP que debe resolver en este proyecto se encuentran en el sitio web `http://www.uv.es/corberan/instancias.htm`. Las instancias también están disponibles en la página web del curso. Su programa debe ser capaz de cargar las instancias en el formato que ellas poseen. Como podrá observar, una de las informaciones que proporciona el sitio web es el valor óptimo de la solución de cada instancia.

8. Informe del proyecto

Debe realizar un informe con el siguiente contenido:

1. Portada.
2. Diseño de la solución.
3. Detalles de la implementación.
4. Resultados experimentales.
5. Análisis de los resultados.
6. Conclusiones.
7. Referencias (en caso de tenerlas).

En la sección de resultados debe presentar 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.

En el informe también debe indicar los detalles de la plataforma usada para realizar los experimentos, esto es, (I) el sistema de operación, (II) la versión del compilador de `Kotlin`, (III) la versión de la maquina virtual de `Java`, (IV) el modelo del CPU, y (V) la cantidad de memoria del computador usado.

9. Condiciones de entrega

Los códigos del proyecto, el informe y la declaración de autenticidad debidamente firmada, deben estar contenidos en un archivo comprimido, con formato *tar.xz*, llamado *Proy2_X_Y.tar.xz*, donde X y Y son los número de carné de los estudiantes. La entrega del archivo *Proy2_X_Y.tar.xz*, debe hacerse por medio de la plataforma *Classroom* antes de las 12:00 pm del día domingo 16 de enero de 2022.

Referencias

[1] David Avis. A survey of heuristics for the weighted matching problem. *Networks*, 13(4):475–493, 1983.

[2] Enrique Benavent, Vicente Campos, Angel Corberán, and Enrique Mota. Analisis de heuristicos para el problema del cartero rural. *Trabajos de estadística y de investigación operativa*, 36(2):27–38, 1985.

[3] N. Christofides, V. Campos, A. Corberan, and E. Mota. An algorithm for the rural postman problem. *Imperial College Report IC-OR-81-5*, 81, 1981.

[4] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.

[5] HA Eiselt, M. Gendreau, and G. Laporte. Arc Routing Problems, Part II: The Rural Postman Problem. *Operations Research*, 43(3):399–414, 1995.

[6] R.S. Garfinkel and I.R. Webb. On crossings, the crossing postman problem, and the rural postman problem. *Networks*, 34(3):173–180, 1999.

[7] Vladimir Kolmogorov. Blossom v: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1):43–67, 2009.

[8] Wen Lea Pearn and TC Wu. Algorithms for the rural postman problem. *Computers & Operations Research*, 22(8):819–828, 1995.

Guillermo Palma / gvpalma@usb.ve / 07 de enero de 2022.