

Tarea N° 1

Ruteo de Vehículos con Caminatas

Integrantes: Michelle Contreras

Noelia Falcuk

Luca Zanella

Profesores: Gonzalo Muñoz

Fernando Ordóñez

Auxiliar: Daniel Hermosilla

Heurísticas para Optimización Entera y Aplicaciones

Código: IN7605

Fecha de entrega: 29 de septiembre de 2025

1. Metodología

Resumen

La elaboración de nuestra heurística se basó en confeccionar dos métodos distintos de construcción y mejora de caminos, incorporando componentes aleatorios, para luego combinarlos simplemente quedándonos con el mejor camino devuelto por alguno de los métodos. En resumen, la heurística final consiste de la combinación de dos heurísticas distintas para encontrar dos rutas distintas y seleccionar la mejor entre las dos que surgen de la *heurística 1* y la *heurística 2*.

Heurística 1

La confección del primer método tiene como primer paso determinar los posibles puntos pickup P_j para cada trabajador j . En particular,

$$P_j = \{\text{nodo } v : d(w_j, v) \leq r_j\}$$

donde $d(\cdot, \cdot)$ denota la distancia más corta en G de nodo a nodo, es el conjunto de todos los nodos a los que el trabajador j puede llegar desde su casa w_j tal que camina a lo más r_j .

Luego, mantenemos un conjunto *uncovered* de todos los trabajadores sin recoger, los cuales al inicio (nodo 0) son todos los trabajadores de una instancia. Ahora, desde el nodo actual, v_i , consideramos todos los nodos que son alcanzables a través de algún camino (en particular, consideramos el camino más corto a tal nodo) y chequeamos cuántos trabajadores sin recoger podrían tener a ese nodo como punto de pickup usando P_j para cada trabajador j . En este punto tendremos una lista de posibles nodos a los cuales nos podemos mover desde v_i . El próximo paso es generar un puntaje $S(v)$ para cada nodo $v \in G$ alcanzable desde v_i , tal que

$$S(v) = \frac{|\{j \in \text{uncovered} : v \in P_j\}|}{d(v_i, v)}$$

Ahora, ordenamos todos estos puntajes de mayor a menor. La solución naïve sería movernos directamente al nodo con el mejor puntaje. Sin embargo, para explorar rutas que no son óptimas inmediatamente decidimos escoger a cuál nodo movernos de manera aleatoria, siempre y cuando nuestro puntaje se mantenga dentro de cierto rango. Este proceso es conocido como GRASP con un restricted candidate list (RCL), como lo vimos en la auxiliar número 2. Existen distintas formas de construir nuestra RCL, en nuestro caso usaremos un α especificado por el usuario. Este α nos hace incluir en la RCL a los nodos cuyo puntaje esté a lo más $\alpha\%$ alejado del mejor puntaje (notamos que $\alpha = 0$ corresponde a la solución naïve o golosa, y $\alpha = 1$ es puramente aleatoria). Para nuestra heurística, tomamos $\alpha = 0.2$ y luego escogemos un nodo uniformemente al azar de la RCL para decidir a dónde movernos. Una vez escogido el nodo, lo añadimos a la ruta y quitamos todos los trabajadores que tienen a ese nodo como posible punto pickup de los no recogidos y repetimos el proceso a partir del nodo seleccionado. Una vez que el número de trabajadores no recogidos sea cero, tenemos una lista de puntos de pickup que cubren a todos los trabajadores y que por construcción están conectados en el orden que fueron agregados. Sin embargo, es posible que ese orden no sea el óptimo, así que implementamos una mejora local de 2-opt para intercambiar dos puntos pickup e invertir el orden de los nodos entre ellos (sujeto a factibilidad). Finalmente, repetimos este proceso de GRASP+2-opt por un número de iteraciones especificadas por el usuario. Una vez tengamos el

orden final de los puntos de pickup encontramos los caminos más cortos que los unen y devolvemos la ruta con los nodos intermedios.

El pseudocódigo de nuestra primera heurística se encuentra en Algoritmo 1

Heurística 2

La confección del segundo método tiene su eje en la decisión glotona de considerar que los nodos "más importantes" para incorporar al camino serán aquellos que cubran más trabajadores. Partiendo de dicha intuición, lo que hicimos fue elegir un umbral aleatorio $k \in [3, M]$, con M la máxima cantidad de trabajadores cubiertos por algún nodo, y luego seleccionar a todos los nodos que cubran a más de k trabajadores para que formen parte del camino.

Luego, mirando los trabajadores que no fueron cubiertos por el primer conjunto de nodos seleccionados, vamos iterando el siguiente proceso: elegir uno de los trabajadores no cubiertos, seleccionar los nodos que lo cubren, y ordenarlos por score de mayor a menor, en donde el score de un nodo es la cantidad de trabajadores no cubiertos que cubre. Luego, de entre los $j = 10$ nodos con mejor score, se elige uno de forma aleatoria y se lo incorpora al conjunto de nodos elegidos para formar parte del camino. Se recalcula la lista de trabajadores no cubiertos, y se repite el proceso hasta que todos los trabajadores hayan sido cubiertos.

A continuación, para armar el camino propiamente, se inicia por el nodo 0, y se selecciona, del conjunto de nodos que formarán parte de nuestro camino, a aquél con menor distancia al 0. En segundo lugar, se elige al siguiente nodo con menor distancia al nodo seleccionado en la primera posición, y así sucesivamente hasta terminar de ordenar los nodos.

Como la lista ordenada de nodos no representa un camino, se realiza un proceso de "costura", en el que se va completando el camino entre cada par de nodos de la lista utilizando el camino mínimo entre dicho par de nodos. Al finalizar el proceso, tenemos un camino factible y que cubre a todos los trabajadores.

Algoritmo 1: Heurística 1

Entrada: Un grafo pesado dirigido $G = (V, E)$ y una lista W de workers (w_i, r_i) tales que el worker i queda cubierto por un nodo v si $d(w_i, v) < r_i$.

Resultado: Una ruta que parte del nodo 0 y retorna al mismo nodo, cubriendo todos los trabajadores.

```

1 Para  $j \in W$  hacer
2   |   Calcular sus puntos pickup, i.e.,  $P_j$ 
3 fin
4 Inicializamos lista de trabajadores sin cubrir,  $uncovered = W$ 
5 Inicializamos puntos intermedios de la ruta  $R$  Nodo actual,  $v_i = 0$ 
6 Mientras  $uncovered \neq \emptyset$  hacer
7   |   Para  $v \in G$  hacer
8     |     Chequear que existe un camino desde  $v_i$  a  $v$ 
9     |     Si  $d(v_i, v) \neq \infty$  entonces
10    |       |   Calcular  $S(v)$ 
11    |       |   Añadir  $S(v)$  a una lista  $scores$ 
12    |   fin
13   | fin
14   |    $best =$  mejor puntaje en  $scores$ 
15   |   Crear  $RCL$  con aquellos puntajes que sean  $\geq best \cdot (1 - \alpha)$ 
16   |   Tomar un puntaje de  $RCL$  al azar
17   |   Set  $v_i$  como el nodo con el que obtuvimos ese puntaje
18   |   Añadir  $v_i$  a  $R$ 
19   |   Quitar de  $uncovered$  a todos  $w \in W$  cubiertos por  $v_i$ 
20 fin
21  $R' =$  Nuevo orden al hacer  $2 - opt$  con los  $v \in R$ 
22 Ruta = Encontrar el camino más corto que une a los  $v \in R'$ 
23 return Ruta

```

Algoritmo 2: Heurística 2

Entrada: Un grafo pesado dirigido $G = (V, E)$ y una lista W de workers (w_i, r_i) tales que el worker i queda cubierto por un nodo v si $d(w_i, v) < r_i$.

Resultado: Una ruta que parte del nodo 0 y retorna al mismo nodo, cubriendo todos los trabajadores.

- 1 Inicializamos umbral $k = randint(3, M)$;
- 2 Inicializamos $S = \{\text{nodos que cubren a más de } k \text{ workers}\}$,
 $\text{uncovered} = \{\text{workers no cubiertos por } S\}$;
- 3 **Mientras** $\text{uncovered} \neq \emptyset$ **hacer**
- 4 Tomar $w \in \text{uncovered}$;
- 5 Seleccionar cjto. A de nodos v que cubren a w y asignarles score
 $s = \{w_j \in \text{uncovered} : d(v, w_j) < r_j\}$;
- 6 Elegir uniformemente al azar un v de entre los 10 de A con mejor score y agregarlo a S ;
- 7 Sacar de uncovered los workers que hayan sido cubiertos por v ;
- 8 **fin**
- 9 Ordenar S por distancias mínimas parciales en una lista ruta, comenzando en 0 y terminando en 0 ;
- 10 Coser ruta agregando el camino mínimo entre cada par de nodos ;
- 11 **return** ruta ;

3. Metodología de la variante

La heurística explicada previamente nos entrega inicialmente una ruta que atiende a todos los trabajadores. Sobre esa solución de referencia, la variante propuesta consiste en analizar si conviene no atender a algunos de ellos.

La estrategia implementada es la siguiente:

1. Se parte de la mejor ruta obtenida con la heurística original, que cubre a todos los trabajadores.
2. Se elige un número r de trabajadores a excluir. Para cada valor de r se generan aleatoriamente distintas combinaciones de exclusión, evitando evaluar el espacio completo de combinaciones que sería exponencial.
3. Para cada combinación de trabajadores excluidos:
 - Se bloquean sus nodos (hogar y nodos cubiertos).
 - Se reconstruye la ruta factible en el grafo resultante.
 - Se calcula el nuevo costo de esa ruta.
 - Se cuenta cuántos trabajadores quedaron realmente sin cobertura.
 - Se calcula la métrica

$$P = \frac{\text{costo original} - \text{costo nuevo}}{\text{número de trabajadores excluidos}},$$

que representa la mejora de costo promedio por trabajador no atendido.

4. Finalmente se selecciona la combinación con mayor valor de P , lo que indica que la exclusión de ese subconjunto de trabajadores es la que resulta más conveniente desde el punto de vista de la empresa.

De esta manera, el método aprovecha la solución original como punto de partida, pero permite explorar mejoras en el costo global a cambio de excluir a ciertos trabajadores, resolviendo así la variante del problema.

Algoritmo 3: Evaluación de exclusión de trabajadores

Entrada: Un grafo pesado dirigido $G = (V, E)$ con matriz de distancias D , lista W de workers (w_i, r_i), matriz de cobertura C , ruta inicial ruta y costo mejor_costo.

Resultado: Valores (P^*, r^*, u^*) donde P^* es la máxima mejora por trabajador excluido, r^* el tamaño de la combinación y u^* los no cubiertos correspondientes.

```

1 Inicializar  $P^* = 0, r^* = 0, u^* = 0;$ 
2 Para  $r \leftarrow 1$  to  $r_{max}$  hacer
3   Generar todas las combinaciones  $\mathcal{C}$  de  $r$  workers;
4   Si  $r > 1$  y  $|\mathcal{C}| > 50$  entonces
5     | Seleccionar al azar 50 combinaciones de  $\mathcal{C}$ ;
6   fin
7   foreach  $comb \in \mathcal{C}$  do
8     ruta'  $\leftarrow$  copia(ruta);
9      $B \leftarrow$  nodos bloqueados de workers en  $comb$  (incluye  $w_i$ , excluye el depósito 0);
10    Eliminar de ruta' los nodos en  $B$ ;
11    Construir  $D'$  y  $G'$  anulando nodos en  $B$ ;
12    nueva_ruta  $\leftarrow []$ ;
13    Para  $(u, v)$  pares consecutivos en ruta' hacer
14      Si  $D'[u, v] = \infty$  entonces
15        | Coser camino mínimo entre  $u$  y  $v$  en  $G'$ ;
16        Si no existe camino entonces
17          | saltar a siguiente combinación;
18        fin
19        Agregar camino cosido a nueva_ruta;
20      Sino
21        | Agregar  $(u, v)$  a nueva_ruta;
22      fin
23    fin
24    Si nueva_ruta es válida (ciclo desde 0) entonces
25      costo'  $\leftarrow \sum D'[x, y] \forall (x, y) \in$  nueva_ruta;
26       $u \leftarrow$  número de workers no cubiertos en nueva_ruta;
27      Si  $u > 0$  entonces
28        |  $P \leftarrow (\text{mejor\_costo} - costo')/u$ ;
29        Si  $P > P^*$  entonces
30          | |  $P^* \leftarrow P, r^* \leftarrow r, u^* \leftarrow u$ ;
31        fin
32      fin
33    fin
34  end
35 fin
36 return  $(P^*, r^*, u^*)$ ;

```

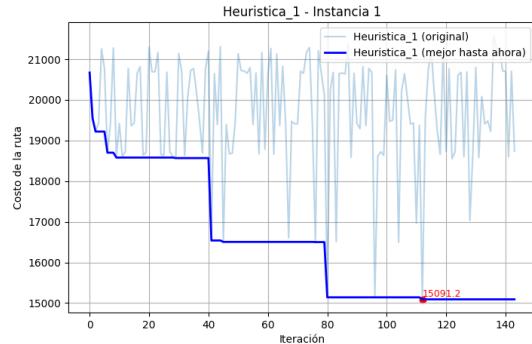
4. Resultados

Evolución de las heurísticas

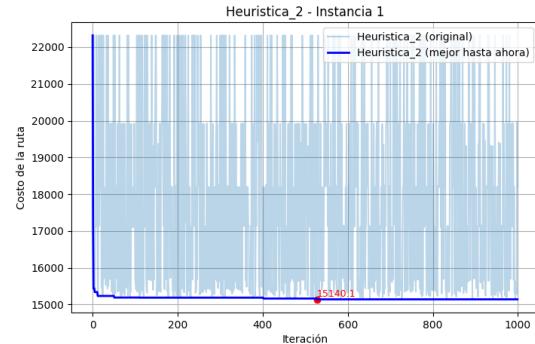
En nuestro setup, usamos Python 3.11, con semilla 123456 y un límite de 30 segundos por heurística. Luego, tomamos el mejor resultado de las dos heurísticas y seleccionamos esa ruta. Evaluamos la evolución de cada método manteniendo un historial de todos los costos explorados y seguimos el costo más bajo alcanzado hasta cada iteración.

Observamos que la *heurística 2* tiende a converger más rápido que la *heurística 1* en algunas instancias, ya que actúa de manera glotonía. Por otra parte, la *heurística 1* puede no escoger soluciones tan buenas inicialmente, debido a la aleatoriedad al escoger un nodo de la RCL.

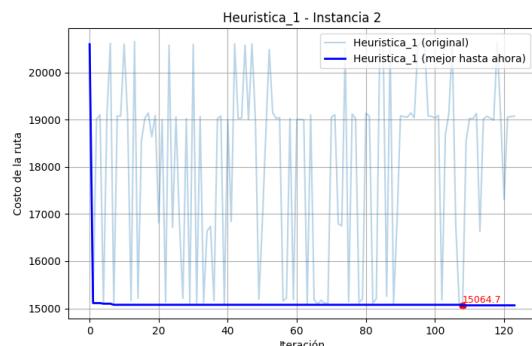
Comparación de costos y gráficos de rutas óptimas



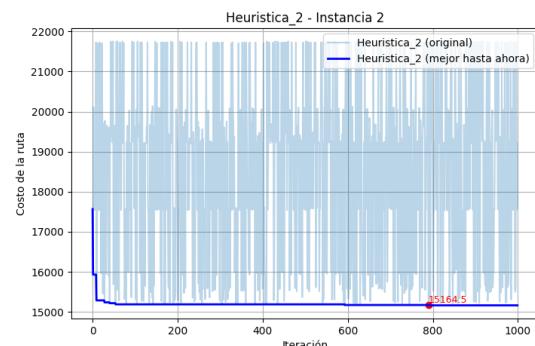
(a) Heurística 1 - Instancia 1



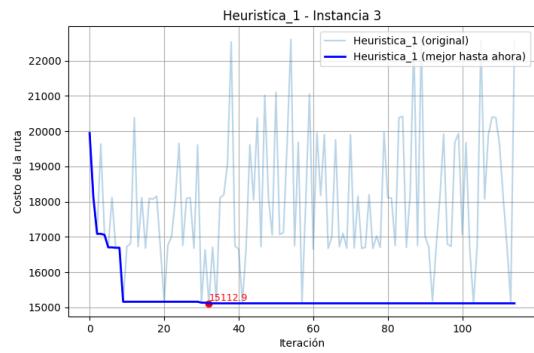
(b) Heurística 2 - Instancia 1



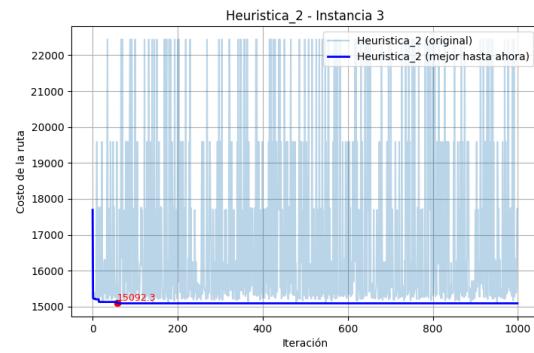
(a) Heurística 1 - Instancia 2



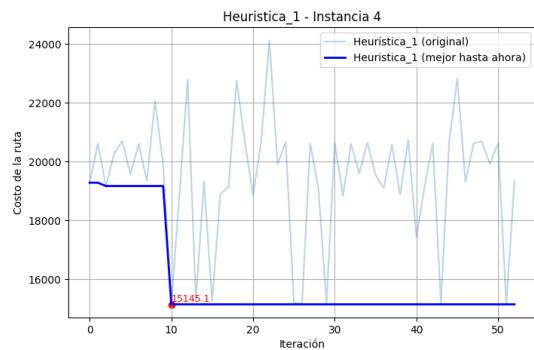
(b) Heurística 2 - Instancia 2



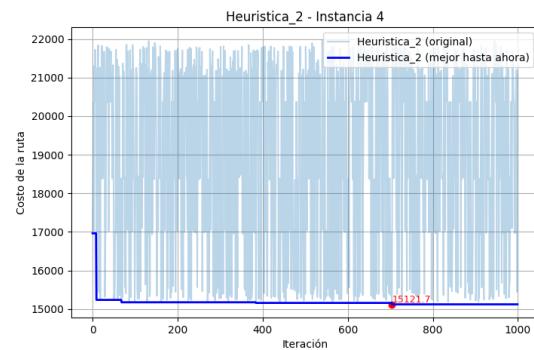
(a) Heurística 1 - Instancia 3



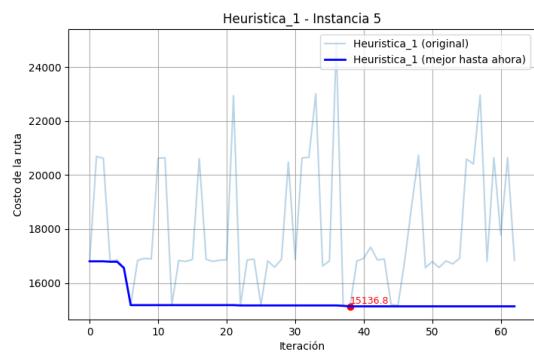
(b) Heurística 2 - Instancia 3



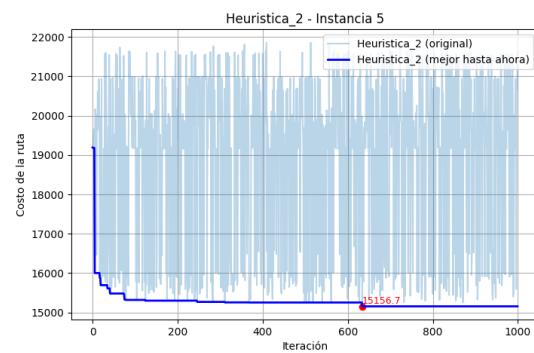
(a) Heurística 1 - Instancia 4



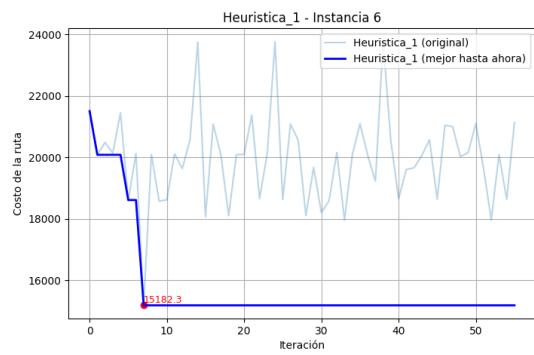
(b) Heurística 2 - Instancia 4



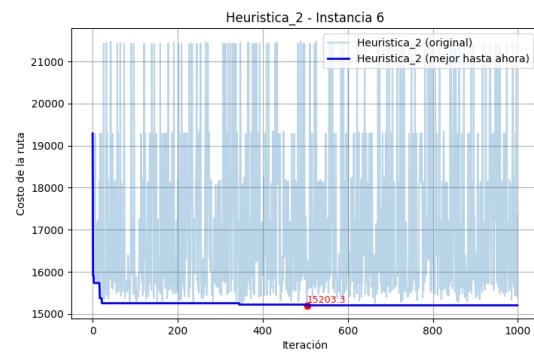
(a) Heurística 1 - Instancia 5



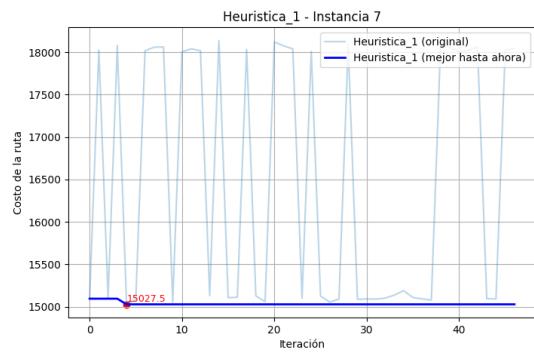
(b) Heurística 2 - Instancia 5



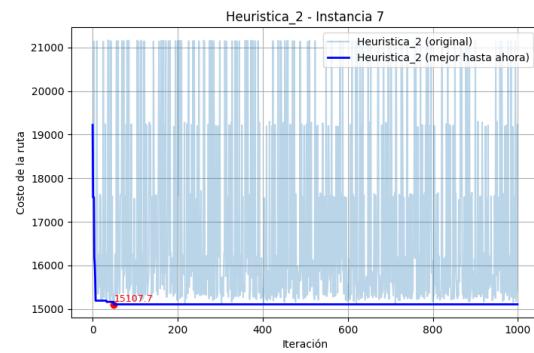
(a) Heurística 1 - Instancia 6



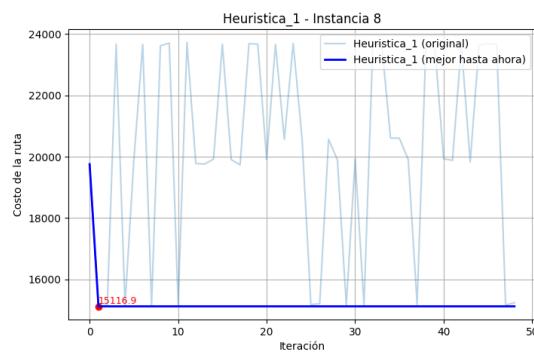
(b) Heurística 2 - Instancia 6



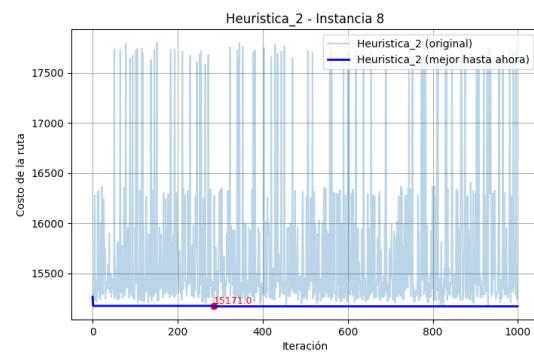
(a) Heurística 1 - Instancia 7



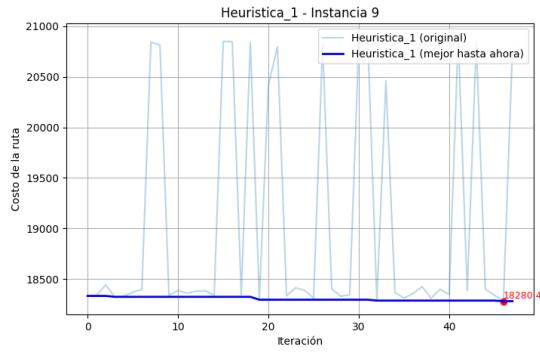
(b) Heurística 2 - Instancia 7



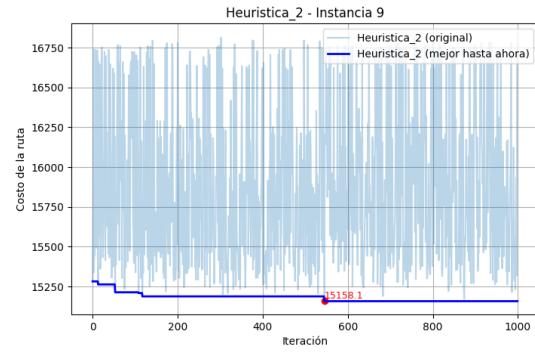
(a) Heurística 1 - Instancia 8



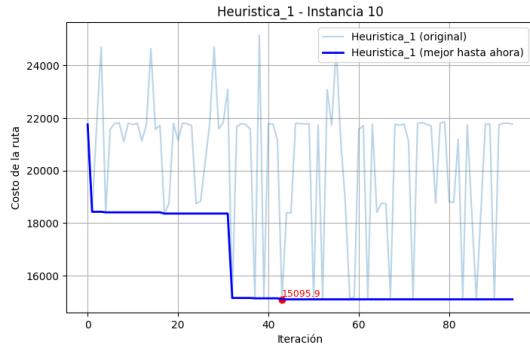
(b) Heurística 2 - Instancia 8



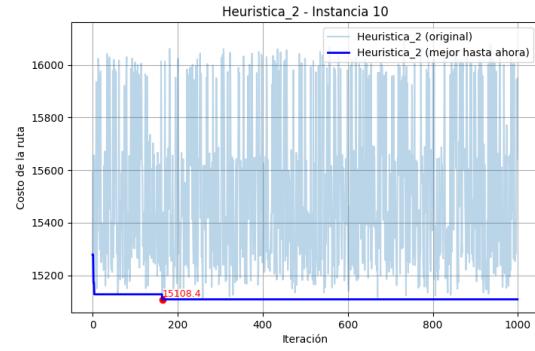
(a) Heurística 1 - Instancia 9



(b) Heurística 2 - Instancia 9



(a) Heurística 1 - Instancia 10



(b) Heurística 2 - Instancia 10

Instancia	Heurística 1	Heurística 2	Ganadora
1	15091.2	15140.1	H1
2	15064.7	15164.5	H1
3	15112.9	15092.3	H2
4	15145.1	15121.7	H2
5	15136.8	15156.7	H1
6	15182.3	15203.3	H1
7	15027.5	15107.7	H1
8	15116.9	15171.0	H1
9	18280.4	15158.1	H2
10	15095.9	15108.4	H1

Tabla 5.1: Tabla comparativa de mejores costos alcanzados por cada heurística

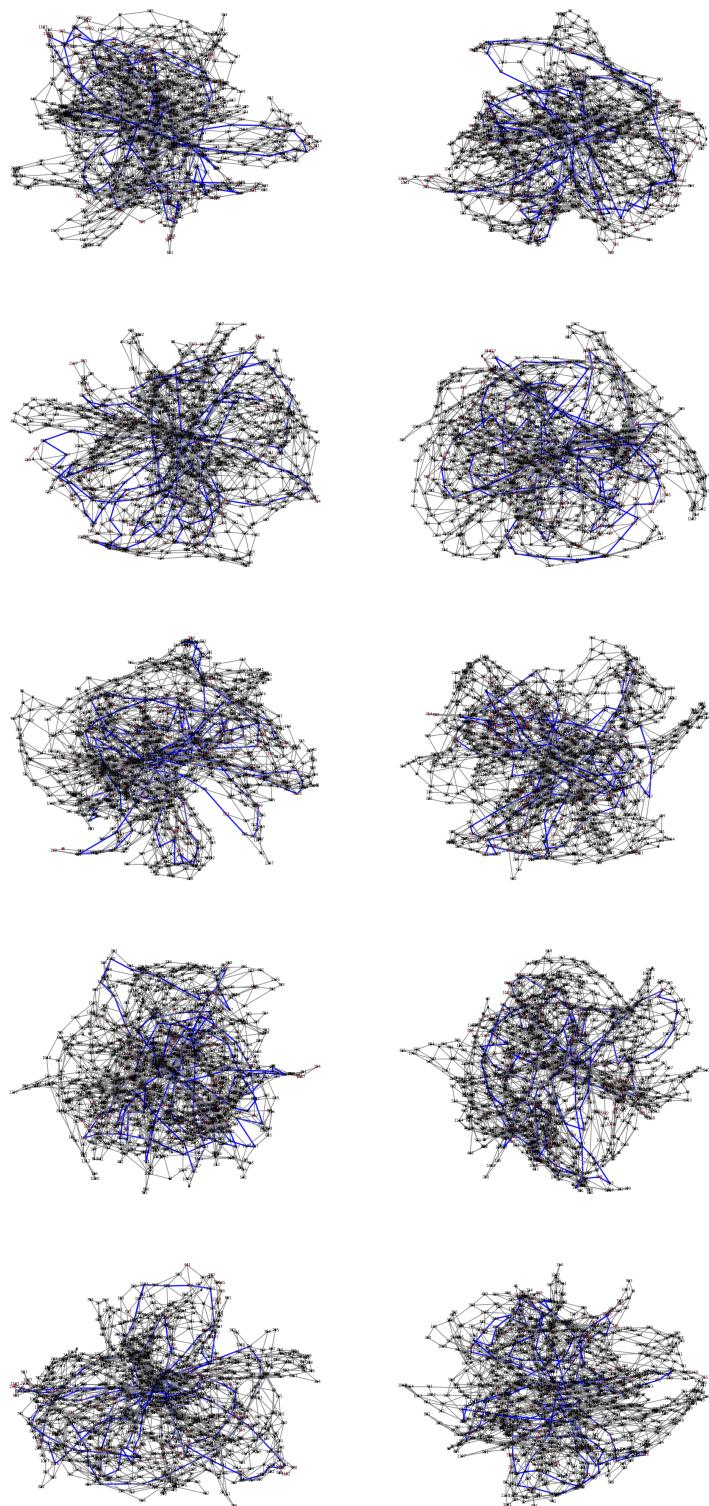


Figura 5.11: Rutas óptimas ordenadas por instancia