Proyecto Final: Problema de Cartero Chino Múltiple

Fernanda Flores[1] Diego García Tinajero[1] Adrián Martínez[1] Aldo Rangel [1]
Jannet Tamayo[1] Iván Vega[1]

[1]Centro de Investigación en Matemáticas A.C.
Avenidad de la plenitud 103, Fracc. José Vasconcelos
CP 20200, Aguascalientes, Ags.

Resumen

En el presente trabajo se aborda el problema del cartero chino múltiple, el cual pertenece a los problemas de ruteo por arcos (Arc Routing Problem). El objetivo del trabajo es proponer una heurística capaz de obtener soluciones factibles que minimicen el costo de las rutas en tiempos cortos.

I. Introducción

Los problemas de ruteo por arcos mejor conocidos en inglés como Arc Routing Problems (ARP) tienen su origen en el famoso problema los siete puentes de Konigsberg resuelto por Leonhard Euler en 1736. Estos problemas se suelen plantear como la búsqueda de la ruta óptima que recorre todas o un subconjunto de las aristas de un grafo dado.

Los tres principales problemas ARP son:

- Problema del cartero chino: se busca hacer un recorrido que minimice el costo y que se recorran todas las aristas del grafo al menos una vez.
- Problema del cartero rural: es una generalización del problema del cartero chino, la diferencia radica en que solo se debe visitar un subgrupo de las aristas del grafo.
- Capacitated arc routing problem:es una generalización de los dos anteriores, sin embargo, en este caso se permite que más de un vehículo haga el recorrido.

El problema del cartero chino (Chinese Postman Problem CPP) fue planteado por el matemático chino Guan Mei-Ko, sin embargo, fue Alan Goldman quien decidió llamarlo "el problema del cartero chino". El planteamiento de Guan Mei-Ko era el problema al que se enfrentaba un cartero para repartir la correspondencia recorriendo la menor distancia posible, es decir, consiste en encontrar un tour en el grafo con el menor costo.

Una variante del problema del cartero chino es el problema de los k carteros (k-CPP), básicamente consiste en considerar que en lugar de tener un cartero se tienen k carteros, lo que se traduce a encontrar k tours y que la suma de los costos de estos k tours sea mínima.

Actualmente este tipo de problemas tienen una aplicación en problemas reales, tales como el mantenimiento de calles y principalmente en logística urbana, sin embargo, también es aplicable en otras

áreas como la producción de circuitos electrónicos integrados o la asignación de tareas. Debido al impacto de los problemas de ruteo por arcos, éstos han sido ampliamente estudiados, en particular, en este trabajo se plantea una heurística para obtener soluciones del problema de los k carteros. Los resultados que se obtuvieron fueron para 10 instancias para los casos de considerar 1, 2, 3 y 4 carteros.

II. Marco teórico

El problema del cartero chino requiere que la grafo en cuestión G sea no dirigida, y que toda arista tenga asociado un costo no negativo y sea visitada al menos una vez en la ruta de costo mínimo. Dos resultados importantes a tener en cuenta son:

1. Un grafo G es par (es decir, que el grado de todo vértice es par) si y sólo si contiene un tour euleriano, es decir, un camino que pasa por cada arista una y solo una vez.

2. En un grafo no dirigida G se cumple que el número de nodos de grado impar es par.

Nótese que en un grafo que cumple 1, la solución óptima es la suma de los costos de las aristas. En el caso de tener nodos impares, el grafo se transforma par duplicando aristas y construyendo un tour euleriano sobre la gráfica modificada, lo que resultará en la ruta de coste mínimo en grafo original.

Como ya se mencionó, una extensión del problema es el k-CPP, en donde en lugar de buscar una ruta se buscan k rutas, todos con el mismo nodo de partida, y que entre las k rutas se atraviesen todas las aristas.

Los problemas de ruteo sobre arcos han sido ampliamente estudiados. Existen distintas formulaciones para resolver, por ejemplo, en [3] se aplican algoritmos genéticos. Para este caso, sin embargo, se hace uso de algoritmos tipo UMDa.

III. Metodología

Empezando por el caso cuando solo se tiene un cartero, en un principio es necesario obtener los nodos impares para posteriormente determinar las conexiones entre estos. Para representar estas conexiones se utiliza un vector del tamaño del número de los nodos impares, donde el nodo en la primera posición se conecta con el de la segunda posición, el nodo de la tercera posición con el nodo de la cuarta posición y así sucesivamente (ver figura 1), esta estructura representará las soluciones dentro del algoritmo de optimización.

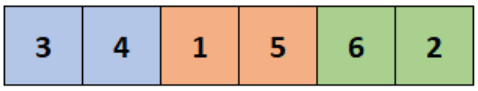

Figura 1. Ejemplo de representación de las conexiones de los nodos impares. El nodo 3 se conecta con 4, el 1 con 5 y el 6 con 2.

La función objetivo es minimizar el coste de la ruta, de acuerdo a la teoría vista en el marco teórico, el coste de la mejor ruta se calcula sumando los costos de las aristas en el grafo más la distancia mínima entre los nodos impares que se conectaron. Al trabajar con estos grafos se utilizó la librería Networkx, entre las bondades de esta librería se encuentra la obtención de la distancia o camino más

corto entre 2 nodos haciendo uso del algoritmo de Dijkstra. Así pues, para minimizar la función objetivo es necesario encontrar la mejor combinación de conexiones de nodos impares.

Las soluciones se obtendrían utilizando un algoritmo heurístico tipo UMDa, el cual utiliza un vector de parámetros y los individuos de la población se crean a través del muestreo. Primeramente, se tiene una población inicial aleatoria, que en nuestro caso es de tamaño 100, donde cada vector individuo es una permutación de los nodos impares, tal como se ejemplificó en la Figura 1, y se calcula el valor objetivo de cada elemento de la población. De este modo, el valor objetivo del individuo es la suma del total de aristas del gráfico máslas distancias entre las parejas que define.

Para que la distancia entre cada par de nodos impares sea mínima, se crea un matriz de distancia con el algoritmo de Dijkstra entre los nodos impares. Al tener la población inicial ya evaluada, se seleccionan los mejores 10 individuos, es decir, los de menor valor objetivo. Posteriormente se crean dos vectores de parámetros de tamaño del número de nodos impares, que contienen la media y desviación estándar respectivamente para cada variable. A partir de estos vectores se hace muestreo usando la distribución normal para generar 100 vectores del tamaño de la solución, los índices de este vector se ordenan de menor a mayor, lo cual define las permutaciones de nodos impares de la nueva población. Posteriormente se evalua cada individuo como ya se explicó, y se repite la selección y el muestreo. Este proceso se repite un número de 1000 iteraciones.

Cuando se tienen $n > 1$ carteros, el problema se divide en dos partes (ver figura 2). La primera parte es asignar las aristas correspondientes a cada uno de los carteros, y la segunda parte es resolver cada uno de los carteros de manera individual. El coste total es la suma del coste de la ruta de cada uno de los carteros.

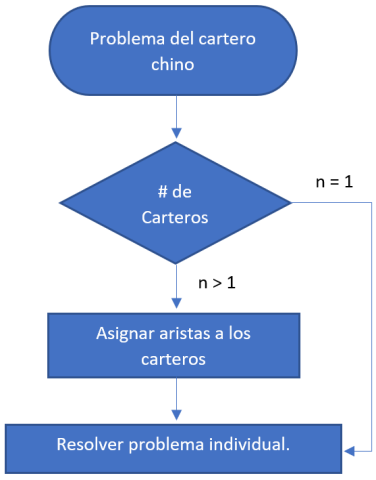

Figura 2. Algoritmo general para la solución del problema del cartero Chino Múltiple

Para asignar las aristas primero resolvemos el problema de dividir el grafo en n grupos, minimizando el coste de los cortes de las aristas. Para resolver esto utilizamos un algoritmo heurístico tipo UMDA, similar al utilizado para resolver las conexiones de los nodos impares. En este caso para representar los cortes utilizamos un vector con las permutaciones de los nodos, este es dividido en partes iguales de acuerdo al número de cortes, así pues cada una de estas partes representa cada grupo de nodos los cuales se asignarán a cada cartero. Los pasos de selección y muestreo son iguales a como

ya se ha explicado.

Una vez hecho el corte que minimiza el coste de las aristas cortadas, cada grupo de nodos y sus aristas son asignados a un cartero. Las aristas cortadas en el algoritmo anterior no se están tomando en cuenta, por lo que es necesario asignarlas a alguno de los carteros. Para realizar esta asignación para cada cartero se obtiene qué nodos necesitaría para poder considerar en su ruta las aristas faltantes, siempre y cuando la arista faltante tenga conexión directa a uno de los nodos de este cartero. Para cada nodo que necesita un cartero se calcula el peso de estos el cual es el número de aristas faltantes en los cuales involucra este nodo. Una vez hecho este calculo, se asigna el nodo con mayor peso al cartero correspondiente, y esto se repite hasta que ya no existan aristas sin considerar.

En el ejemplo de la figura 3, observamos dos grupos de nodos, cada uno para un cartero. La línea negra muestra las 3 aristas que son cortadas debido a esta asignación de nodos. En la primera iteración, el nodo marcado con 'a' lo necesita el grupo de nodos de color rojo para dos aristas faltantes, por lo que tiene peso = 2. Los nodos marcados con 'c', 'd', 'e' son necesarios para el grupo de nodos de color verde, cada uno con peso 1 y de igual manera el nodo 'b' pero para el grupo rojo. Por lo tanto el nodo 'a' es asignado al grupo rojo en la primera iteración. En la siguiente iteración solo queda una arista que no es tomada en cuenta, entonces los nodos 'b' y 'e' tienen peso = 1 para los grupos rojo y verde respectivamente, al tener el mismo peso se asigna de manera aleatoria, en este caso el grupo verde se queda con el nodo 'e'. Finalmente todas las aristas se han tomado en cuenta y se procede a resolver las conexiones entre nodos impares para cada grupo de manera individual.

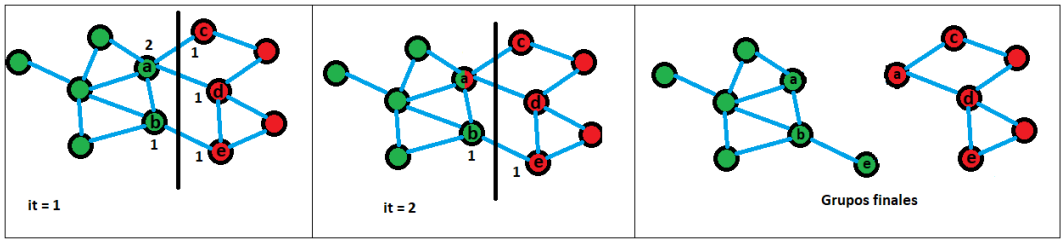

Figura 3. Algoritmo para asignar aristas faltantes a los carteros

IV. Experimentación y resultados

Una vez determinados ambos modelos tanto para el caso individual como para el múltiple, se realizaron 50 corridas con $k = 1000$ iteraciones para todas las instancias y para $n = 1, 2, 3, 4$ carteros, en un equipo de cómputo con un procesador Intel(R) Core(TM) i5-1135G7 a 2.40GHz con 12 gb de memoria RAM.

Para cada instancia y un número de carteros determinados de las 50 corridas se obtuvieron su media, desviación estándar y la mejor solución encontrada con el algoritmo propuesto. En la Tabla 1 se muestran dichos datos. Podemos observar que a medida que se aumentó el número de carteros creció la desviación estándar dentro de la muestra y también aumentó el valor de la función objetivo en la mejor solución encontrada.

Instancia	Número de carteros	Media	Desviación estándar	Mejor solución
val1A	1	173	0	173
	2	187.12	5.691526104	172
	3	194.72	10.29174426	174
	4	203.36	9.654775703	176
val2A	1	217.02	0.141421356	217
	2	263.08	12.29541146	246
	3	298.96	20.71572409	259
	4	313.12	19.90040508	278
val3A	1	77	0	77
	2	88.6	3.057276366	84
	3	98.92	6.298169478	87
	4	107.42	5.095376055	96
val4A	1	388	0	388
	2	411.08	9.623186286	392
	3	423.72	10.9563885	402
	4	464.54	12.58637099	444
val5A	1	416.14	1.958862644	415
	2	452.42	10.52516906	413
	3	483.18	17.16152555	447
	4	519	21.78559719	471
val6A	1	221.28	0.701019666	221
	2	242.78	5.46711878	234
	3	249.3	9.926772704	227
	4	248.08	7.339688032	235
val7A	1	296.54	2.296492445	288
	2	301.38	5.461890231	294
	3	317.72	9.583233658	298
	4	336.2	10.18602484	312
val8A	1	385.06	0.239897937	385
	2	441.86	9.231733205	424
	3	473.06	12.64815869	446
	4	508.56	23.04313612	460
val9A	1	324.96	2.258408325	323
	2	345.48	5.866682127	332
	3	362.74	6.945296456	349
	4	382.56	8.261442282	365
val10A	1	426.16	1.461897006	425
	2	464.54	6.021085399	451
	3	496.68	12.66272047	468
	4	507.8	10.76653925	487

Tabla 1. Resultados obtenidos de la experimentación con el algoritmo propuesto

En la Figura 4 podemos observar nuevamente el aumento del valor de la mejor solución conforme aumenta el número de carteros.

Después se hizo un análisis del tiempo medio que tomó cada corrida del algoritmo nuevamente por cada instancia y cada número de carteros, la Tabla 2 nos muestra los datos obtenidos. Es evidente que al aumentar el número de carteros, aumenta a su vez la complejidad del problema y por ende también el tiempo que toma resolverlo. En la Figura 5 se observa que el mayor salto en el tiempo de cómputo se da en el momento en el que el problema se vuelve múltiple y posteriormente a medida que incrementa el número de carteros asignados, la tasa de crecimiento es menor.

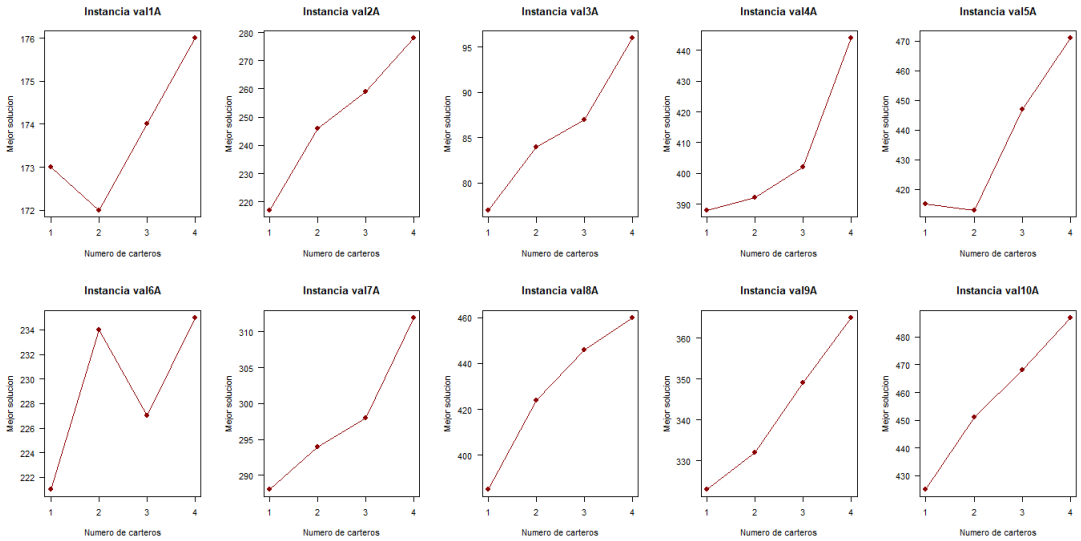

Figura 4. Comparación de soluciones

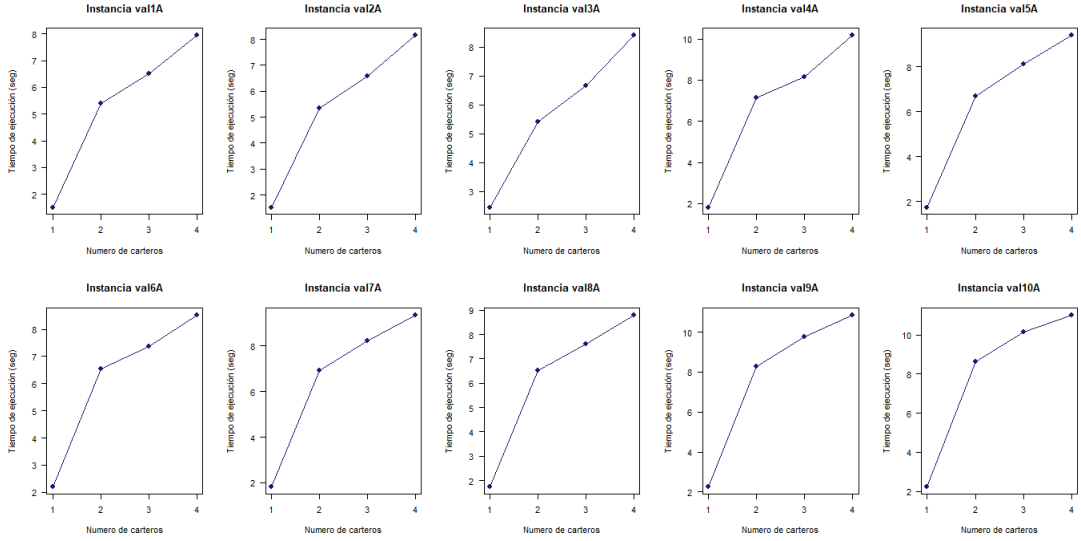

Figura 5. Comparación del tiempo computacional.

Instancia	Número de carteros	Tiempo promedio de ejecución (seg)
val1A	1	1.505029302
	2	5.406379271
	3	6.492179203
	4	7.943661213
val2A	1	1.518937385
	2	5.33757062
	3	6.572320032
	4	8.154090357
val3A	1	2.448887956
	2	5.401081657
	3	6.663828039
	4	8.408615828
val4A	1	1.810480618
	2	7.154637718
	3	8.158266068
	4	10.18009377
val5A	1	1.724215126
	2	6.692608643
	3	8.124600506
	4	9.402527237
val6A	1	2.194348168
	2	6.539580441
	3	7.352308369
	4	8.531206608
val7A	1	1.809000611
	2	6.914208937
	3	8.236834097
	4	9.366900253
val8A	1	1.734089208
	2	6.522339487
	3	7.618604469
	4	8.805728912
val9A	1	2.249228179
	2	8.282748222
	3	9.769453955
	4	10.86449728
val10A	1	2.224965334
	2	8.617328882
	3	10.16748762
	4	11.02863712

Tabla 2. Tiempo promedio de ejecución del algoritmo

En (6) se muestra la representación gráfica de la instancia val1A, se puede observar que hay 24 nodos. Además, dentro de la instancia se asigna al nodo 1 como depósito. Por otro lado, suponiendo que se tienen 3 carteros, en las figuras (7), (8) y (9), se muestran los recorridos para los carteros 1, 2 y 3, respectivamente. Notemos que como bien se comentó en la metodología y se visualizó en la figura (2) algunos nodos se incluyen en más de un subgrafo, asimismo, el depósito se considera para los carteros 1 y 2, sin embargo, a pesar que el depósito no aparece en el recorrido del cartero 3, sí se toma en cuenta en la función objetivo del algoritmo, simplemente se suma la arista con el menor costo que se conecta al depósito, en este caso sería sumar el peso de la arista que conecta a los nodos 1 y 11.

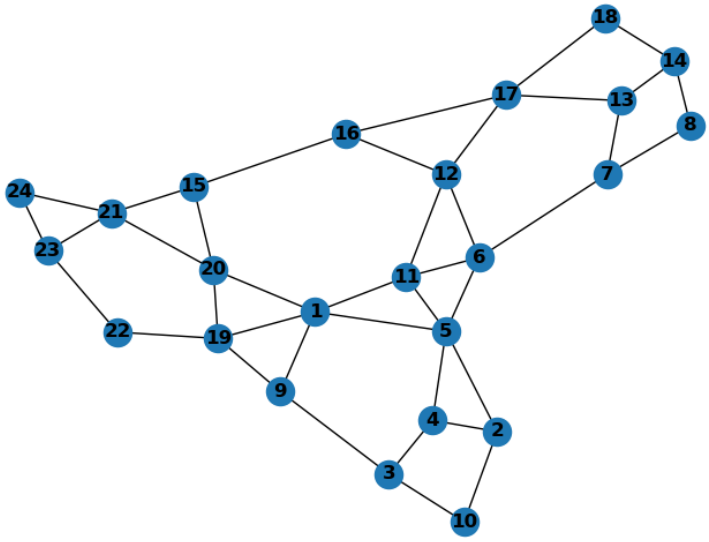

Figura 6. Recorrido completo

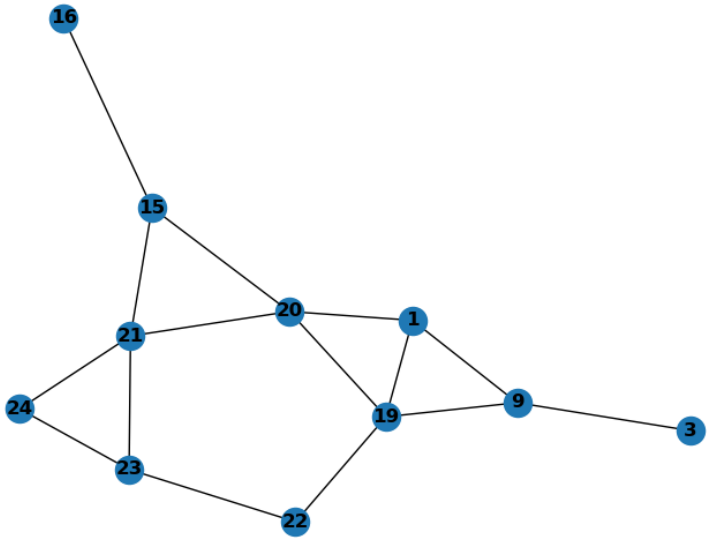

Figura 7. Recorrido del cartero 1

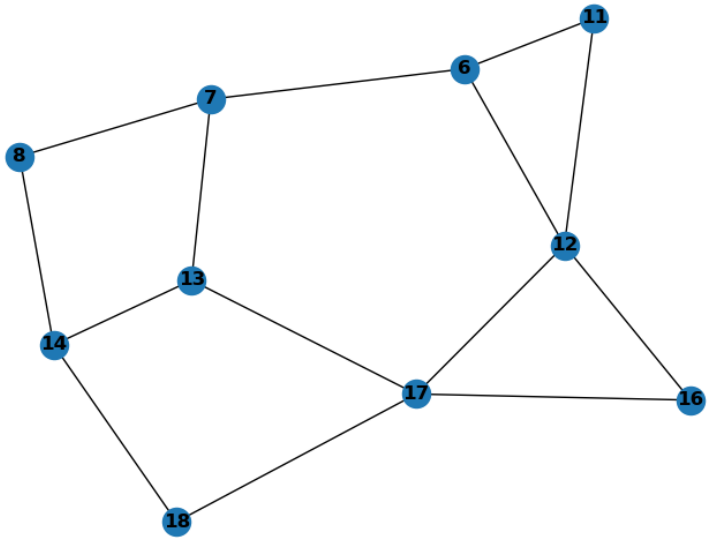

Figura 8. Recorrido del cartero 2

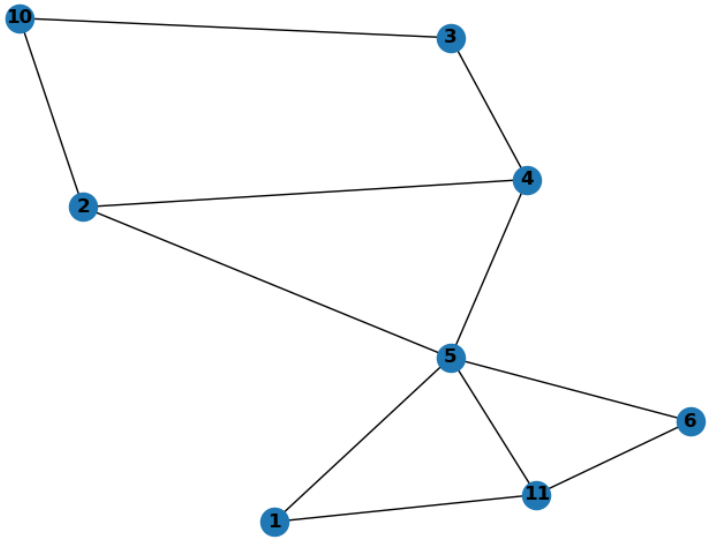

Figura 9. Recorrido del cartero 3

V. Conclusiones

Como se observa en los resultados, al considerar sólo minimizar las distancias totales de pasar por cada arista al menos una vez y que el recorrido de cada cartero debe regresar al depot, aumentar el número de carteros no sólo no mejora sino empeora el valor objetivo, y aumenta el tiempo del entorno de ejecución. Una variante del problema que además de estos factores considerara la minimización del tiempo de recorrido o de las horas trabajadas por cartero, podría verse beneficiado de aumentar el número de éstos.

VI. Bibliografía

Referencias

[1] Holmberg, K. (2018). Heuristics for the weighted k-rural postman problem with applications to urban snow removal. Journal on Vehicle Routing Algorithms, 1(2), 105-119.

[2] Corberán, Á., Laporte, G. (Eds.). (2015). Arc routing: problems, methods, and applications. Society for Industrial and Applied Mathematics.

[3] Hua, J., Li-Shan, K. (2002). Genetic Algorithm for Chinese Postman Problems. State Key Laboratory of Software Engineering, Wuhan University. Hubei, China.

[4] Grado (teoría de grafos). (2021, 15 de octubre). Wikipedia, La enciclopedia libre. Fecha de consulta: 12:32, junio 14, 2022 desde https://es.wikipedia.org/wiki/https://es.wikipedia.org/wiki/Grado_(teoria_de_grafos).

[5] Yordá, J. (2014). El problema del cartero chino [Tesis de Máster]. Universidad de Santiago de Compostela.

[6] Algoritmo de Dijkstra. (2022, 6 de enero). Wikipedia, La enciclopedia libre. Fecha de consulta: 13:40, junio 14, 2022 desde https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra