CUANDO EL MÁS CERCANO NO ES EL PREFERIBLE

Alfredo G. Hernández Díaz
Profesor Catedrático, Universidad Pablo de Olavide, España
Ana D. López-Sánchez
Profesora Contratada Doctora, Universidad Pablo de Olavide, España
Jesús Sánchez-Oro
Profesor Contratado Doctor, Universidad Rey, España
Abraham Duarte
Profesor Catedrático, Universidad Rey Juan Carlos, España

RESUMEN

En este trabajo presentamos un problema de localización de instalaciones que aparece en múltiples situaciones reales y que, por tanto, es de gran interés para diversos sectores. Generalmente, cuando se desea localizar un conjunto de centros logísticos o de distribución que van a prestar un servicio tanto a empresas minoristas como mayoristas, se intenta que estén situados lo más cerca posible de estas empresas.

Además, se suele asumir que cada empresa recibirá el servicio del centro logístico más cercano. Esto se traduce en que el objetivo o criterio a optimizar sea situar los centros logísticos en aquellos lugares que minimicen la máxima distancia entre las empresas y su centro logístico, asegurando así que la empresa más alejada de su centro de distribución estará lo más cerca posible y, por tanto, todas las demás empresas recorrerán una distancia inferior.

Sin embargo, aparecen situaciones en las cuales algún o algunos centros logísticos podrían quedar inhabilitados debido, por ejemplo, a una catástrofe natural. En este caso, las empresas que son servidas por los centros logísticos afectados tendrán que recibir el servicio desde otro punto. Este otro centro podría ser el segundo más cercano o, si éste también estuviera no disponible, desde el tercero más cercano o, si éste tampoco pudiese dar servicio, desde el cuarto más cercano, y así sucesivamente. Este problema no ha sido muy estudiado en la literatura, pero cada vez se aplica más en situaciones reales propias, por ejemplo, de la Logística Humanitaria. Debido a su complejidad, los métodos exactos no permiten resolver este problema en tiempos razonables.

Por ello, proponemos un algoritmo metaheurístico capaz de resolverlo rápidamente obteniéndose soluciones de gran calidad. Concretamente, se propone un algoritmo GRASP combinado con una Búsqueda Tabú y una Oscilación Estratégica.

1. INTRODUCCIÓN

Existen muchas situaciones reales en las que hay que instalar un conjunto de instalaciones (hospitales, centros comerciales, estaciones de bicicleta, centros logísticos, etc.) para dar un servicio a un conjunto de puntos de demanda (pacientes, clientes, usuarios de bicicleta, empresas, etc.). Hay multitud de problemas de localización según la función que se desee optimizar, algunos de los más conocidos son el problema del p-centro, el problema del p-mediana o el problema de la máxima cobertura. Una característica común que la mayoría de los problemas de localización tienen es que cada punto de demanda será servido por su instalación más cercana ya que así se reduce el coste o el tiempo en el que se obtiene el servicio.

Sin embargo, podría ocurrir que algunas de las instalaciones se queden inhabilitadas debido a algún fallo y que haya que acudir a otra instalación para que preste el servicio requerido por los puntos de demanda. Podemos distinguir entre dos situaciones, que no se conozca de antemano qué instalaciones están inhabilitadas o que si se conozcan. En el primer caso, el cliente debe acudir a la instalación para poder comprobar que no puede obtener el servicio requerido, y una vez allí se dirigirá a la siguiente instalación más cercana a ésta. Este problema se conoce como el problema del próximo p-centro (Albareda-Sambola et al., 2015 o López-Sánchez et at., 2019) y el objetivo es minimizar la máxima distancia entre los puntos de demanda y las instalaciones más cercanas mas la distancia entre las instalaciones y las instalaciones más cercanas a éstas. En el segundo caso, el cliente sabe a priori qué instalaciones no están disponibles y en lugar de ir la instalación más cercana, si ésta no estuviese disponible, se dirigirá directamente a la segunda o si esta no estuviera disponible, se dirigirá a la tercera, y así sucesivamente. Este problema se conoce como el problema del $\propto$-ésimo p-centro (Chen y Chen, 2013) y el objetivo es minimizar la máxima distancia entre los puntos de demanda y la $\propto$-ésima instalación más cercana. En este trabajo se resolverá esta versión del problema, el $\propto$-ésimo p-centro que como se ha indicado su objetivo es localizar un conjunto p de instalaciones, seleccionadas de un conjunto m de posibles instalaciones candidatas, con $p<m$, para servir a un conjunto de n puntos de demanda de forma que se minimice la máxima distancia entre cada punto de demanda y su $\propto$-ésima instalación más cercana, para que todos los puntos de demandas puedan tener, no sólo una única instalación cercana sino $\propto$ instalaciones cercanas asegurando que en el caso de que alguna instalación quedase inhabilitada, poder dirigirnos a la siguiente y que ésta siga estando cerca.

Para abordar el problema se ha diseñado un algoritmo metaheurístico basado en un algoritmo GRASP (Festa y Resende, 2009) para construir soluciones y mejorarlas usando en lugar de una búsqueda local tradicional, una búsqueda Tabú (Glover y Laguna, 1998) Estas soluciones se incluyen en un algoritmo de Oscilación Estratégica (Glover, 2000).

2. ALGORITMO PROPUESTO

Como se introdujo anteriormente el problema del $\propto$-ésimo p-centro pretende: minimizar la máxima distancia entre cada punto de demanda y su $\propto$-ésima instalación más cercana. Teniendo este objetivo, a continuación, se detalla el algoritmo diseñado para obtener soluciones para este problema.

2.1 GRASP

Para construir soluciones factibles usaremos una metodología GRASP (Festa y Resende, 2009) que es un algoritmo multi-arranque con dos fases: fase de construcción, que es la encargada de obtener soluciones factibles desde cero y fase de mejora, que es la encargada de mejorar las soluciones obtenidas en la fase de construcción.

En la fase de construcción partimos de una solución vacía y se elige aleatoriamente la primera instalación que se va a localizar y a continuación se ordenan en una lista el resto de las instalaciones basándonos en una medida de contribución que dependerá de la función a optimizar. Este algoritmo en lugar de seleccionar siempre la mejor instalación elegirá aleatoriamente una instalación de las mejores instalaciones hasta que se haya construido una solución completa, es decir, hasta que se hayan seleccionado p instalaciones. Esto se repite N veces para tener una población inicial de soluciones.

Una vez construidas N soluciones factibles y como es usual en un algoritmo GRASP, se intentarán mejorar las soluciones aplicándoles una búsqueda local. Una búsqueda local sencilla consistiría simplemente en intercambiar una instalación que haya sido incluida en la solución con otra que no lo haya sido y ver si somos capaces de mejorarlas haciendo este simple intercambio. Sin embargo, en esta fase en lugar de optar por una búsqueda tradicional hemos implementado una búsqueda tabú (Glover y Laguna, 1998) puesto que uno de los inconvenientes de la búsqueda local tradicional es que la mejora de la solución depende fuertemente de la solución de partida, pudiendo quedar fácilmente en un óptimo local. La búsqueda tabú tiene dos características importantes, memoria adaptativa y aceptación o no de movimientos. En este caso, dispondremos de una memoria a corto plazo que almacenará los últimos movimientos que se hayan realizado, de esta forma no podrán repetirse movimientos mientras que estén almacenados en la memoria, con el fin de restringir los movimientos disponibles y se permitirá empeorar el valor de la función objetivo para evitar quedar atrapado en un óptimo local. La búsqueda tabú termina cuando no se encuentren mejoras después de un número determinado de iteraciones.

2.2 Oscilación estratégica

Las soluciones obtenidas en la fase anterior serán incluidas en un procedimiento de oscilación estratégica (Glover, 2000) que consiste en explorar nuevas regiones del espacio de soluciones gracias a movernos entre la factibilidad y la infactibilidad de las soluciones.

Partimos de una solución factible obtenida en la Sección 2.1, y añadiremos las δ instalaciones para obtener soluciones infactibles con $p + \delta$ instalaciones. A continuación, para recuperar la factibilidad se eliminan δ instalaciones. Se han seguido los siguientes criterios para incluir y eliminar instalaciones:

Insertar	**Eliminar**
Voraz	Voraz
Voraz	Aleatoriamente
Aleatoriamente	Voraz
Aleatoriamente	Aleatoriamente

Tabla 1: Estrategias para insertar y eliminar instalaciones en la oscilación estratégica.

3. RESULTADOS

En esta sección se muestran los resultados que se han obtenido al resolver 37 problemas del repositorio TSP-lib (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html) con el algoritmo propuesto y hemos realizado una comparación con el estado del arte. Concretamente se ha resuelto el problema para $\alpha = 1{,}2{,}3$, siendo un total de 111 instancias. En las tablas 2, 3 y 4 se muestran los valores medios de la función objetivo (FO) y el tiempo de ejecución en segundos (CPU) para cada problema solucionado. Se muestra una comparativa de nuestro algoritmo propuesto denominado (SO) y el estado del arte (SOTA) y además se incluyen los límites inferiores (LB) para cada problema.

Como puede observarse en las tablas 2, 3 y 4, el algoritmo propuesto presenta un comportamiento robusto como mostraron las 30 ejecuciones independientes que se realizaron, independientemente del valor de $\propto$ seleccionado.

	LB		SOTA		SO	
Instance	**FO**	**CPU (s)**	**FO**	**CPU (s)**	**FO**	**CPU (s)**
att48_48_10	836.34	0.09	1401.71	0.09	1203.18	1.85
att48_48_20	474.65	0.11	921.43	0.11	710.77	0.65
att48_48_30	251.07	0.08	921.43	0.08	462.08	0.22
att48_48_40	186.95	0.07	331.05	0.07	319.85	0.06
ch150_150_10	122.55	8.59	177.64	8.59	141.53	82.96
ch150_150_100	18.91	1.78	45.25	1.78	33.47	1.50
ch150_150_110	16.48	1.83	48.22	1.83	30.18	0.95
ch150_150_120	13.77	1.73	40.72	1.73	27.36	0.55
ch150_150_130	11.91	1.76	33.40	1.76	22.45	0.26
ch150_150_140	8.80	1.46	36.68	1.46	17.58	0.11
ch150_150_20	76.80	14.33	133.37	14.33	97.13	39.97
ch150_150_30	55.72	9.55	118.75	9.55	79.56	18.01
ch150_150_40	47.41	6.14	93.49	6.14	68.23	12.22
ch150_150_50	38.01	3.94	79.66	3.94	60.94	7.74
ch150_150_60	31.98	3.95	68.02	3.95	49.64	6.29
ch150_150_70	27.06	2.11	68.02	2.11	46.48	4.38
ch150_150_80	24.99	1.72	49.64	1.72	41.46	3.28
ch150_150_90	20.73	1.83	45.95	1.83	38.38	2.18
eil101_101_10	12.14	2.72	18.68	2.72	14.32	26.04
eil101_101_100	0.71	0.18	1.41	0.18	1.41	0.05
eil101_101_20	7.53	5.78	12.73	5.78	10.30	8.50
eil101_101_30	5.76	2.06	12.04	2.06	8.25	4.75
eil101_101_40	4.61	1.16	11.18	1.16	7.28	2.89
eil101_101_50	3.64	1.00	10.82	1.00	7.07	1.80
eil101_101_60	3.35	0.65	7.62	0.65	6.32	1.08
eil101_101_70	2.69	0.35	7.21	0.35	5.00	0.55
eil101_101_80	2.24	0.28	6.71	0.28	4.12	0.25
eil101_101_90	1.58	0.20	6.32	0.20	3.16	0.08
pr439_439_10	1716.51	42.85	2580.94	42.85	1971.83	3790.85
pr439_439_20	1029.71	120.01	2958.57	120.01	1200.26	1566.51
pr439_439_30	739.19	279.80	1630.38	279.80	886.71	761.30
pr439_439_40	580.01	319.76	1530.52	319.76	728.87	490.00
pr439_439_50	468.54	368.89	1570.03	368.89	600.00	294.52
pr439_439_60	400.20	370.46	1654.73	370.46	548.29	230.09
pr439_439_70	357.95	271.50	1630.38	271.50	500.00	175.29
pr439_439_80	312.50	264.24	1630.38	264.24	475.66	155.71
pr439_439_90	280.90	93.01	1144.83	93.01	416.08	131.18

Tabla 2: Resultados computacionales para $\alpha = 1$

	LB		SOTA		SO	
Instance	**FO**	**CPU (s)**	**FO**	**CPU (s)**	**FO**	**CPU (s)**
att48_48_10	1377.53	0.20	2334.17	0.20	1592.12	5.14
att48_48_20	820.45	0.22	1910.06	0.22	1130.85	1.21
att48_48_30	601.59	0.22	1849.16	0.22	936.38	0.42
att48_48_40	474.65	0.19	1267.89	0.19	532.08	0.07
ch150_150_10	184.06	0.95	241.53	0.95	205.66	223.16
ch150_150_100	38.01	3.61	91.90	3.61	53.21	2.35
ch150_150_110	33.24	3.46	77.13	3.46	51.65	1.36
ch150_150_120	31.98	3.20	77.13	3.20	50.30	0.72
ch150_150_130	29.55	2.49	87.22	2.49	46.63	0.31
ch150_150_140	27.06	2.23	71.82	2.23	42.30	0.14
ch150_150_20	122.55	13.20	194.37	13.20	141.53	94.75
ch150_150_30	93.98	26.27	161.82	26.27	112.51	55.58
ch150_150_40	76.38	20.94	118.83	20.94	96.42	31.74
ch150_150_50	65.42	19.11	166.24	19.11	87.69	18.10
ch150_150_60	55.72	8.02	129.57	8.02	78.42	12.24
ch150_150_70	50.78	5.97	117.58	5.97	68.23	8.20
ch150_150_80	47.41	4.22	98.11	4.22	64.45	5.57
ch150_150_90	41.28	4.14	103.50	4.14	62.04	3.63
eil101_101_10	19.46	0.45	25.94	0.45	21.21	68.12
eil101_101_100	3.64	0.68	6.32	0.68	2.83	0.05
eil101_101_20	12.14	7.98	19.10	7.98	14.14	28.27
eil101_101_30	9.22	7.22	18.68	7.22	12.00	10.63
eil101_101_40	7.53	9.60	15.13	9.60	9.43	6.19
eil101_101_50	6.36	4.05	15.81	4.05	8.60	3.19
eil101_101_60	5.76	2.85	13.60	2.85	8.25	1.94
eil101_101_70	5.00	1.61	13.60	1.61	7.28	0.96
eil101_101_80	4.53	1.43	12.04	1.43	6.32	0.43
eil101_101_90	4.03	1.06	9.43	1.06	5.00	0.11
pr439_439_10	2752.64	0.37	3779.05	0.81	3146.63	12960.97
pr439_439_20	1716.51	0.84	2440.54	26.27	2226.26	6158.89
pr439_439_30	1271.83	0.94	3222.19	24.36	1500.21	3982.18
pr439_439_40	1008.17	3.40	3300.00	76.85	1253.99	2587.24
pr439_439_50	874.27	4.53	2432.33	183.53	1068.00	1327.63
pr439_439_60	739.19	7.90	2037.31	487.18	975.00	918.13
pr439_439_70	621.74	25.81	2575.12	1271.12	905.54	639.50
pr439_439_80	580.01	24.80	2277.06	497.79	731.86	509.87
pr439_439_90	530.48	49.18	2432.33	291.78	715.89	405.64

Tabla 3: Resultados computacionales para $\alpha = 2$

	LB		SOTA		SO	
Instance	**FO**	**CPU (s)**	**FO**	**CPU (s)**	**FO**	**CPU (s)**
att48_48_10	1965.63	0.01	2755.07	0.01	2186.31	6.72
att48_48_20	1179.16	0.35	2011.66	0.33	1374.48	1.61
att48_48_30	829.38	0.54	1628.44	0.50	1011.66	0.54
att48_48_40	676.75	0.22	1374.48	0.22	675.00	0.08
ch150_150_10	295.81	0.15	330.21	0.16	298.56	398.03
ch150_150_100	52.60	5.76	109.42	7.79	69.35	3.23
ch150_150_110	48.80	6.83	134.29	8.85	67.22	1.85
ch150_150_120	47.41	4.79	108.78	5.18	61.29	0.95
ch150_150_130	42.59	4.62	104.00	4.43	57.50	0.41
ch150_150_140	40.01	4.82	77.13	4.33	52.20	0.16
ch150_150_20	168.41	19.58	203.04	13.83	179.71	150.94
ch150_150_30	122.55	26.00	211.07	35.26	146.41	78.08
ch150_150_40	103.43	25.38	167.28	25.21	119.22	52.10
ch150_150_50	87.13	26.36	161.82	27.68	108.03	26.70
ch150_150_60	76.38	21.82	164.50	26.16	97.46	17.78
ch150_150_70	69.37	31.32	134.29	30.88	92.82	13.10
ch150_150_80	61.58	17.73	131.63	17.82	83.38	8.34
ch150_150_90	55.72	9.74	133.65	12.64	79.81	4.75
eil101_101_10	30.22	0.11	33.24	0.10	29.43	92.44
eil101_101_100	5.17	2.24	10.20	2.15	2.83	0.05
eil101_101_20	16.32	2.70	20.81	3.41	18.03	43.73
eil101_101_30	12.14	4.77	16.28	5.61	14.14	19.37
eil101_101_40	10.31	10.51	21.40	11.04	12.04	9.71
eil101_101_50	9.01	6.60	21.54	7.03	10.63	4.74
eil101_101_60	7.53	8.05	13.60	8.47	9.06	2.22
eil101_101_70	6.71	8.09	14.42	8.72	8.54	1.06
eil101_101_80	6.08	4.06	15.13	4.49	7.28	0.44
eil101_101_90	5.76	2.50	10.77	2.95	6.08	0.11
pr439_439_10	3989.30	0.49	4601.22	0.09	4076.23	16272.47
pr439_439_20	2347.51	6.27	3220.64	29.76	2726.03	11827.09
pr439_439_30	1716.51	2.01	3390.06	18.51	2231.73	5679.48
pr439_439_40	1407.62	3.21	3222.19	67.48	1644.88	4102.42
pr439_439_50	1226.02	3.23	3486.58	38.89	1467.35	3039.05
pr439_439_60	1019.99	5.76	2622.26	105.76	1340.01	2033.03
pr439_439_70	946.46	17.48	3222.19	102.37	1231.11	1316.50
pr439_439_80	853.03	54.00	3390.06	402.48	1217.58	955.74
pr439_439_90	739.19	21.18	2982.03	491.31	986.47	723.38

Tabla 4: Resultados computacionales para $\alpha = 3$

4. CONCLUSIONES

En este trabajo hemos abordado un problema de optimización conocido como el problema $\propto$-ésimo p-centro y que aparece cuando fallan las $\propto -1$-ésimas instalaciones más cercanas teniendo que ir a la $\propto$-ésima instalación más cercana. Aquí se han abordaddo situaciones con $\propto = 1,2,3$ puesto que en situaciones reales suele ser improbable que fallen más de 3

instalaciones. El algoritmo que se propone es capaz de obtener soluciones independientemente del valor de $\propto$ que se indique y esa flexibilidad es una de las fortalezas del algoritmo propuesto.

REFERENCIAS

ALBAREDA-SAMBOLA, M., HINOJOSA, Y., MARÍN, A., PUERTO, J. (2015) When centers can fail: A close second opportunity. Computers & Operations Research 62 (Supplement C), pp.145-156

CHEN, D., CHEN, R. (2013) Optimal algorithms for the alpha-neighbor p-center problem. European Journal of Operational Research 225(1), pp. 36-43

FESTA, P.; RESENDE, M.G. (2009) An annotated bibliography of GRASP–Part I: Algorithms. International Transactions in Operational Research 16, pp.1–24

GLOVER, F. (2000) Multi-start and strategic oscillation methods—principles to exploit adaptive memory. En: Computing Tools for Modeling, Optimization and Simulation (ed.) Springer, pp. 1–23

GLOVER, F., LAGUNA, M. (1998) Tabu search. En: Handbook of combinatorial optimization, (ed.) Springer pp.2093-2229.

LÓPEZ-SÁNCHEZ, A., SÁNCHEZ-ORO, J., HERNÁNDEZ-DÍAZ, A. (2019) Grasp and vns for solving thecp-next center problem. Computers & Operations Research 104, pp. 295-303.