Óscar Darias Plasencia
Maximum Diversity Problem
Diseño y Análisis de Algoritmos
12 May 2017

Resultados obtenidos

En este documento, se analizan los resultados obtenidos utilizando la librería diseñada para resolver el problema de diversidad máxima. Los diferentes algoritmos implementados (véase el informe correspondiente para más información), funcionan de forma bastante diferente unos de otros, y es interesante saber cuál de ellos otorga mejores resultados, y qué clase de mejoras son las más adecuadas para este problema.

Veremos los algoritmos implementados uno por uno, con sus respectivas variantes, rellenando una serie de tablas con los resultados obtenidos. Si es preciso, se llevará a cabo un análisis y una comparativa con los resultados de otros algoritmos.

Algoritmo voraz constructivo

Para el algoritmo voraz constructivo, se han obtenido los diferentes resultados, aplicándolo a las seis instancias del problema de las que disponemos, con diferentes tamaños de solución. Cabe destacar que el tiempo se mide en milisegundos.

Problema	n	K	m	z	S	Tiempo
max_div_15_2	15	2	2	11.859215825677515	[0.58, 1.29] [8.65, 9.98]	0.32
max_div_15_2	15	2	3	25.726199905986398	[9.96, 8.17] [0.58, 1.29] [8.65, 9.98]	0.99
max_div_15_2	15	2	4	48.41393082637548	[9.96, 8.17] [0.58, 1.29] [8.65, 9.98] [1.59, 1.57]	0.86
max_div_15_2	15	2	5	73.56189928048627	[8.41, 9.98] [9.96, 8.17] [0.58, 1.29] [8.65, 9.98] [1.59, 1.57]	1
max_div_20_2	20	2	2	8.510329018316506	[0.63, 5.96] [8.67, 3.17]	0.20
max_div_20_2	20	2	3	21.99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]	0.45
max_div_20_2	20	2	4	39.56822938922611	[1.16, 4.47] [8.57, 8.36] [0.63, 5.96] [8.67, 3.17]	0.43
max_div_20_2	20	2	5	61.23929090001673	[1.16, 4.47] [8.57, 8.36] [8.78, 7.43] [0.63, 5.96] [8.67, 3.17]	0.95
max_div_30_2	30	2	2	11.657143732492965	[9.84, 8.96] [2.07, 0.27]	0.28
max_div_30_2	30	2	3	28.944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]	0.29
max_div_30_2	30	2	4	52.77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]	0.56
max_div_30_2	30	2	5	80.91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]	0.97

Problema	n	K	m	z	S	Tiempo
max_div_15_3	15	3	2	13.27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]	1
max_div_15_3	15	3	3	30.324088261146496	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26]	0.45
max_div_15_3	15	3	4	59.76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]	0.78
max_div_15_3	15	3	5	94.74871918304889	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]	1.2
max_div_20_3	20	3	2	11.800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	0.88
max_div_20_3	20	3	3	30.872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	0.23
max_div_20_3	20	3	4	56.53472607351106	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	0.77
max_div_20_3	20	3	5	92.8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	1.04
max_div_30_3	30	3	2	13.07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]	1.001
max_div_30_3	30	3	3	33.84225896761056	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	0.33
max_div_30_3	30	3	4	63.51841393083194	[6.71, 0.35, 9.42] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	0.24
max_div_30_3	30	3	5	99.50883647734524	[6.71, 0.35, 9.42] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	0.76

Como podemos ver, las medidas del tiempo no han sido demasiado precisas. Los tiempos han tenido que medirse utilizando la función de Java `System.nanoTime()`, cuya precisión no es todo lo recomendable que debería. En cualquier caso, el tiempo de ejecución de estos algoritmos es despreciable: muchas veces ni siquiera llegan al milisegundo.

Algoritmo voraz destructivo

Comprobaremos que los resultados son bastante similares a los obtenidos por el algoritmo voraz constructivo, con algunos casos ligeramente mejores y otros ligeramente peores. En este caso no incluimos una columna con el tiempo de ejecución, puesto que las medidas de este no han sido fieles a la realidad.

Problema	n	K	m	z	S
max_div_15_2	15	2	2	11.859215825677515	[0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	2	3	23.79643771807573	[8.41, 9.98] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	2	4	46.6528110209727	[8.41, 9.98] [0.58, 1.29] [8.65, 9.98] [1.59, 1.57]
max_div_15_2	15	2	5	73.56189928048627	[8.41, 9.98] [9.96, 8.17] [0.58, 1.29] [8.65, 9.98] [1.59, 1.57]
max_div_20_2	20	2	2	8.510329018316506	[0.63, 5.96] [8.67, 3.17]

Problema	n	K	m	z	S
max_div_20_2	20	2	3	21.99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	2	4	39.56822938922611	[1.16, 4.47] [8.57, 8.36] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	2	5	60.43009746590678	[1.16, 4.47] [8.57, 8.36] [8.45, 3.42] [0.63, 5.96] [8.67, 3.17]
max_div_30_2	30	2	2	11.657143732492965	[9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	2	3	28.944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	2	4	52.77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]
max_div_30_2	30	2	5	80.91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]
max_div_15_3	15	3	2	13.27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]
max_div_15_3	15	3	3	30.324088261146496	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26]
max_div_15_3	15	3	4	58.7287294428116	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]
max_div_15_3	15	3	5	96.08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]
max_div_20_3	20	3	2	11.59094905518957	[0.83, 7.06, 3.34] [9.6, 2.02, 9.0]
max_div_20_3	20	3	3	29.15738236020536	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [9.6, 2.02, 9.0]
max_div_20_3	20	3	4	56.690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_20_3	20	3	5	92.8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_30_3	30	3	2	12.613726649963525	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27]
max_div_30_3	30	3	3	34.29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	3	4	63.70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	3	5	99.59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]

Búsqueda local

Para la búsqueda local, lo que haremos será generar primero una solución inicial utilizando el algoritmo voraz destructivo, y luego la mejoraremos. Así, comparando con la tabla anterior, quedará bien claro hasta qué punto es capaz de mejorar la solución.

Problema	n	K	m	z	S
max_div_15_2	15	2	2	11.859215825677515	[0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	2	3	27.372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]

Problema	n	K	m	z	S
max_div_15_2	15	2	4	48.41393082637548	[9.96, 8.17] [0.58, 1.29] [8.65, 9.98] [1.59, 1.57]
max_div_15_2	15	2	5	76.65353336258994	[9.11, 3.23] [8.41, 9.98] [0.58, 1.29] [8.65, 9.98] [1.59, 1.57]
max_div_20_2	20	2	2	8.510329018316506	[0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	2	3	21.99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	2	4	40.00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]
max_div_20_2	20	2	5	62.872862556399284	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [0.63, 5.96] [8.67, 3.17]
max_div_30_2	30	2	2	11.657143732492965	[9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	2	3	28.944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	2	4	52.77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]
max_div_30_2	30	2	5	80.91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]
max_div_15_3	15	3	2	13.27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]
max_div_15_3	15	3	3	31.868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]
max_div_15_3	15	3	4	59.76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]
max_div_15_3	15	3	5	96.08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]
max_div_20_3	20	3	2	11.59094905518957	[0.83, 7.06, 3.34] [9.6, 2.02, 9.0]
max_div_20_3	20	3	3	29.15738236020536	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [9.6, 2.02, 9.0]
max_div_20_3	20	3	4	56.690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_20_3	20	3	5	92.8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_30_3	30	3	2	13.07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]
max_div_30_3	30	3	3	34.29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	3	4	63.70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	3	5	99.59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]

Vemos cómo en unos casos la búsqueda local no logra mejorar la solución. Por ejemplo, para el problema 30_2 para todo tamaño de solución. En otros casos mejora solo unas décimas, como en el problema 30_3 para una solución de tamaño 2. Y otras veces la mejora varios puntos, como en la mayoría de casos con el problema de 15_2.

Algoritmo GRASP

El algoritmo GRASP tiene una serie de parámetros adicionales con respecto a los algoritmos anteriores: el número de iteraciones que se ejecuta y el tamaño de la RCL.

Primero, veamos la tabla para los problemas con K = 2:

Problema	n	m	Iter	|RCL|	z	S
max_div_15_2	15	2	10	2	11.859215825677515	[0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	3	10	2	27.372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	4	10	2	49.54615656742211	[9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	5	10	2	78.18861777882361	[9.11, 3.23] [9.96, 8.17] [0.58, 1.29] [8.65, 9.98] [0.46, 3.05]
max_div_15_2	15	2	20	2	11.859215825677515	[0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	3	20	2	27.372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	4	20	2	49.54615656742211	[9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	5	20	2	78.38849954675047	[9.11, 3.23] [8.41, 9.98] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	2	10	3	11.859215825677515	[0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	3	10	3	27.372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	4	10	3	49.82678080659435	[9.11, 3.23] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	5	10	3	79.12953032810968	[9.11, 3.23] [9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	2	20	3	11.859215825677515	[0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	3	20	3	27.372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	4	20	3	49.54615656742211	[9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]
max_div_15_2	15	5	20	3	79.12953032810968	[9.11, 3.23] [9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]
max_div_20_2	20	2	10	2	8.510329018316506	[0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	3	10	2	21.99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	4	10	2	40.00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]
max_div_20_2	20	5	10	2	63.651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	2	20	2	8.510329018316506	[0.63, 5.96] [8.67, 3.17]

max_div_20_2	20	3	20	2	21.99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	4	20	2	40.00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]
max_div_20_2	20	5	20	2	63.651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	2	10	3	8.510329018316506	[0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	3	10	3	21.99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	4	10	3	39.82918360149513	[3.47, 9.43] [8.57, 8.36] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	5	10	3	63.651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	2	20	3	8.510329018316506	[0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	3	20	3	21.99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]
max_div_20_2	20	4	20	3	40.00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]
max_div_20_2	20	5	20	3	63.651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]
max_div_30_2	30	2	10	2	11.657143732492965	[9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	3	10	2	28.944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	4	10	2	52.77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]
max_div_30_2	30	5	10	2	80.91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]
max_div_30_2	30	2	20	2	11.657143732492965	[9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	3	20	2	28.944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	4	20	2	52.77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]
max_div_30_2	30	5	20	2	80.91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]
max_div_30_2	30	2	10	3	11.657143732492965	[9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	3	10	3	28.944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	4	10	3	52.77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]
max_div_30_2	30	5	10	3	80.91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]
max_div_30_2	30	2	20	3	11.657143732492965	[9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	3	20	3	28.944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]
max_div_30_2	30	4	20	3	52.77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]

max_div_30_2	30	5	20	3	80.91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]

Ahora, las tablas para los problemas con K = 3:

Problema	n	m	Iter	|RCL|	z	S
max_div_15_3	15	2	10	2	13.27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]
max_div_15_3	15	3	10	2	31.868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]
max_div_15_3	15	4	10	2	59.76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]
max_div_15_3	15	5	10	2	96.08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]
max_div_15_3	15	2	20	2	13.27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]
max_div_15_3	15	3	20	2	31.868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]
max_div_15_3	15	4	20	2	59.76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]
max_div_15_3	15	5	20	2	96.08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]
max_div_15_3	15	2	10	3	13.27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]
max_div_15_3	15	3	10	3	31.868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]
max_div_15_3	15	4	10	3	59.76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]
max_div_15_3	15	5	10	3	96.08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]
max_div_15_3	15	2	20	3	13.27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]
max_div_15_3	15	3	20	3	31.868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]
max_div_15_3	15	4	20	3	59.76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]
max_div_15_3	15	5	20	3	96.08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]

Problema	n	m	Iter	|RCL|	z	S
max_div_20_3	20	2	10	2	11.800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]
max_div_20_3	20	3	10	2	30.872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]
max_div_20_3	20	4	10	2	56.690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_20_3	20	5	10	2	92.8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_20_3	20	2	20	2	11.800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]
max_div_20_3	20	3	20	2	30.872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]
max_div_20_3	20	4	20	2	56.690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_20_3	20	5	20	2	92.8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_20_3	20	2	10	3	11.800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]
max_div_20_3	20	3	10	3	30.872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]
max_div_20_3	20	4	10	3	56.690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_20_3	20	5	10	3	92.8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_20_3	20	2	20	3	11.800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]
max_div_20_3	20	3	20	3	30.872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]
max_div_20_3	20	4	20	3	56.690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]
max_div_20_3	20	5	20	3	92.8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]

Problema	n	m	Iter	|RCL|	z	S
max_div_30_3	30	2	10	2	13.07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]
max_div_30_3	30	3	10	2	34.29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	4	10	2	63.70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]

Problema	n	m	Iter	|RCL|	z	S
max_div_30_3	30	5	10	2	99.59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	2	20	2	13.07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]
max_div_30_3	30	3	20	2	34.29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	4	20	2	63.70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	5	20	2	99.59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	2	10	3	13.07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]
max_div_30_3	30	3	10	3	34.29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	4	10	3	63.70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	5	10	3	99.59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	2	20	3	13.07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]
max_div_30_3	30	3	20	3	34.29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	4	20	3	63.70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]
max_div_30_3	30	5	20	3	99.59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]

Como se puede apreciar en los resultados, los números se repiten una y otra vez. Al tratarse de problemas relativamente pequeños, de máximo 30 vectores, realizar 10 iteraciones de un GRASP con una búsqueda local al final de cada una, casi garantiza que se va a obtener la mejor solución, o al menos la mejor solución que cualquier GRASP podría llegar a conseguir. De hecho, en las tablas, observamos que para cualquier número de iteraciones, 10 o 20, y cualquier tamaño de RCL, 2 o 3, los resultados se repiten.

Ramificación y Poda

Para este apartado, veremos diversas tablas en las que se utiliza un determinado algoritmo para generar la cota inferior inicial, y usando diferentes estrategias de exploración del árbol. Vamos a comenzar con una cota inferior suministrada por el algoritmo voraz constructivo, y realizando una exploración primero en profundidad.

Problema	n	K	m	z	S	Tiempo	Nodos generados
max_div_15_2.txt	15	2	2	11,859215825677515	[0.58, 1.29] [8.65, 9.98]	19	44
max_div_15_2.txt	15	2	3	27,372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]	7	56
max_div_15_2.txt	15	2	4	49,82678080659435	[9.11, 3.23] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	20	292
max_div_15_2.txt	15	2	5	79,12953032810968	[9.11, 3.23] [9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	17	217
max_div_20_2.txt	20	2	2	8,510329018316506	[0.63, 5.96] [8.67, 3.17]	14	59
max_div_20_2.txt	20	2	3	21,99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]	8	95
max_div_20_2.txt	20	2	4	40,00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]	11	146
max_div_20_2.txt	20	2	5	63,651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]	30	343
max_div_30_2.txt	30	2	2	11,657143732492965	[9.84, 8.96] [2.07, 0.27]	8	89
max_div_30_2.txt	30	2	3	28,944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]	15	145
max_div_30_2.txt	30	2	4	52,77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]	31	312
max_div_30_2.txt	30	2	5	80,91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]	182	770
max_div_15_3.txt	15	3	2	13,27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]	1	44
max_div_15_3.txt	15	3	3	31,868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]	1	70
max_div_15_3.txt	15	3	4	59,76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]	8	274
max_div_15_3.txt	15	3	5	96,08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]	3	100
max_div_20_3.txt	20	3	2	11,800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	2	59
max_div_20_3.txt	20	3	3	30,872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	3	114
max_div_20_3.txt	20	3	4	56,690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	11	290

max_div_20_3.txt	20	3	5	92,8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	20	404
max_div_30_3.txt	30	3	2	13,07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]	4	89
max_div_30_3.txt	30	3	3	34,29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	4	88
max_div_30_3.txt	30	3	4	63,70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	24	421
max_div_30_3.txt	30	3	5	99,59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	88	1088

Ahora, utilizando también la estrategia de exploración en profundidad, vamos a generar la cota inferior inicial utilizando el algoritmo voraz destructivo.

Problema	n	K	m	z	S	Tiempo	Nodos generados
max_div_15_2.txt	15	2	2	11,859215825677515	[0.58, 1.29] [8.65, 9.98]	26	44
max_div_15_2.txt	15	2	3	27,372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]	7	56
max_div_15_2.txt	15	2	4	49,82678080659435	[9.11, 3.23] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	33	292
max_div_15_2.txt	15	2	5	79,12953032810968	[9.11, 3.23] [9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	12	217
max_div_20_2.txt	20	2	2	8,510329018316506	[0.63, 5.96] [8.67, 3.17]	4	59
max_div_20_2.txt	20	2	3	21,99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]	7	95
max_div_20_2.txt	20	2	4	40,00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]	20	146
max_div_20_2.txt	20	2	5	63,651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]	33	410
max_div_30_2.txt	30	2	2	11,657143732492965	[9.84, 8.96] [2.07, 0.27]	10	89
max_div_30_2.txt	30	2	3	28,944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]	10	145
max_div_30_2.txt	30	2	4	52,77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]	20	312
max_div_30_2.txt	30	2	5	80,91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]	142	770
max_div_15_3.txt	15	3	2	13,27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]	0	44
max_div_15_3.txt	15	3	3	31,868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]	2	70
max_div_15_3.txt	15	3	4	59,76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]	7	286

max_div_15_3.txt	15	3	5	96,08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]	3	100
max_div_20_3.txt	20	3	2	11,800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	2	78
max_div_20_3.txt	20	3	3	30,872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	4	169
max_div_20_3.txt	20	3	4	56,690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	7	290
max_div_20_3.txt	20	3	5	92,8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	15	404
max_div_30_3.txt	30	3	2	13,07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]	6	118
max_div_30_3.txt	30	3	3	34,29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	7	88
max_div_30_3.txt	30	3	4	63,70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	23	421
max_div_30_3.txt	30	3	5	99,59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	90	1088

Por último, probemos con el algoritmo GRASP.

Problema	n	K	m	z	S	Tiempo	Nodos generados
max_div_15_2.txt	15	2	2	11,859215825677515	[0.58, 1.29] [8.65, 9.98]	28	44
max_div_15_2.txt	15	2	3	27,372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]	9	16
max_div_15_2.txt	15	2	4	49,82678080659435	[9.11, 3.23] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	48	292
max_div_15_2.txt	15	2	5	79,12953032810968	[9.11, 3.23] [9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	24	206
max_div_20_2.txt	20	2	2	8,510329018316506	[0.63, 5.96] [8.67, 3.17]	7	59
max_div_20_2.txt	20	2	3	21,99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]	9	95
max_div_20_2.txt	20	2	4	40,00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]	10	92
max_div_20_2.txt	20	2	5	63,651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]	19	225
max_div_30_2.txt	30	2	2	11,657143732492965	[9.84, 8.96] [2.07, 0.27]	28	89
max_div_30_2.txt	30	2	3	28,944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]	19	145
max_div_30_2.txt	30	2	4	52,77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]	87	312
max_div_30_2.txt	30	2	5	80,91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]	104	770

max_div_15_3.txt	15	3	2	13,27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]	2	44
max_div_15_3.txt	15	3	3	31,868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]	1	43
max_div_15_3.txt	15	3	4	59,76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]	8	274
max_div_15_3.txt	15	3	5	96,08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]	4	100
max_div_20_3.txt	20	3	2	11,800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	1	59
max_div_20_3.txt	20	3	3	30,872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	2	114
max_div_20_3.txt	20	3	4	56,690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	9	290
max_div_20_3.txt	20	3	5	92,8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	18	404
max_div_30_3.txt	30	3	2	13,07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]	6	89
max_div_30_3.txt	30	3	3	34,29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	11	88
max_div_30_3.txt	30	3	4	63,70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	31	421
max_div_30_3.txt	30	3	5	99,59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	140	1088

A continuación, vamos a utilizar el algoritmo voraz constructivo para generar la cota inferior inicial, como en el primer caso, solo que esta vez vamos a explorar el árbol expandiendo siempre el nodo con una cota superior más pequeña.

Problema	n	K	m	z	S	Tiempo	Nodos generados
max_div_15_2.txt	15	2	2	11,859215825677515	[0.58, 1.29] [8.65, 9.98]	55	76
max_div_15_2.txt	15	2	3	27,372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]	83	369
max_div_15_2.txt	15	2	4	49,82678080659435	[9.11, 3.23] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	315	1254
max_div_15_2.txt	15	2	5	79,12953032810968	[9.11, 3.23] [9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	2324	4060
max_div_20_2.txt	20	2	2	8,510329018316506	[0.63, 5.96] [8.67, 3.17]	7	166
max_div_20_2.txt	20	2	3	21,99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]	40	458
max_div_20_2.txt	20	2	4	40,00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]	380	1883
max_div_20_2.txt	20	2	5	63,651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]	7444	8436
max_div_30_2.txt	30	2	2	11,657143732492965	[9.84, 8.96] [2.07, 0.27]	18	141

max_div_30_2.txt	30	2	3	28,944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]	43	221
max_div_30_2.txt	30	2	4	52,77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]	747	1866
max_div_30_2.txt	30	2	5	80,91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]	30034	14751
max_div_15_3.txt	15	3	2	13,27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]	1	43
max_div_15_3.txt	15	3	3	31,868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]	18	265
max_div_15_3.txt	15	3	4	59,76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]	79	717
max_div_15_3.txt	15	3	5	96,08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]	453	2004
max_div_20_3.txt	20	3	2	11,800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	5	120
max_div_20_3.txt	20	3	3	30,872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	44	374
max_div_20_3.txt	20	3	4	56,690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	292	1518
max_div_20_3.txt	20	3	5	92,8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	1149	3241
max_div_30_3.txt	30	3	2	13,07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]	20	213
max_div_30_3.txt	30	3	3	34,29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	188	445
max_div_30_3.txt	30	3	4	63,70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	1225	2030
max_div_30_3.txt	30	3	5	99,59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	37970	15929

Ahora repetiremos la misma técnica de exploración, pero generando la cota inferior inicial con el algoritmo voraz destructivo.

Problema	n	K	m	z	S	Tiempo	Nodos generados
max_div_15_2.txt	15	2	2	11,859215825677515	[0.58, 1.29] [8.65, 9.98]	69	76
max_div_15_2.txt	15	2	3	27,372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]	67	335
max_div_15_2.txt	15	2	4	49,82678080659435	[9.11, 3.23] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	458	1468
max_div_15_2.txt	15	2	5	79,12953032810968	[9.11, 3.23] [9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	2200	4060

max_div_20_2.txt	20	2	2	8,510329018316506	[0.63, 5.96] [8.67, 3.17]	6	166
max_div_20_2.txt	20	2	3	21,99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]	39	458
max_div_20_2.txt	20	2	4	40,00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]	396	1883
max_div_20_2.txt	20	2	5	63,651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]	11448	9391
max_div_30_2.txt	30	2	2	11,657143732492965	[9.84, 8.96] [2.07, 0.27]	20	141
max_div_30_2.txt	30	2	3	28,944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]	44	221
max_div_30_2.txt	30	2	4	52,77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]	741	1866
max_div_30_2.txt	30	2	5	80,91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]	27100	14751
max_div_15_3.txt	15	3	2	13,27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]	1	43
max_div_15_3.txt	15	3	3	31,868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]	15	265
max_div_15_3.txt	15	3	4	59,76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]	97	886
max_div_15_3.txt	15	3	5	96,08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]	398	1720
max_div_20_3.txt	20	3	2	11,800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	7	133
max_div_20_3.txt	20	3	3	30,872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	53	475
max_div_20_3.txt	20	3	4	56,690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	322	1453
max_div_20_3.txt	20	3	5	92,8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	1129	3241
max_div_30_3.txt	30	3	2	13,07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]	27	235
max_div_30_3.txt	30	3	3	34,29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	155	445
max_div_30_3.txt	30	3	4	63,70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	1103	2010
max_div_30_3.txt	30	3	5	99,59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	34796	15822

Y por último, utilizando el GRASP:

Problema	n	K	m	z	S	Tiempo	Nodos generados
max_div_15_2.txt	15	2	2	11,859215825677515	[0.58, 1.29] [8.65, 9.98]	137	76
max_div_15_2.txt	15	2	3	27,372700045548008	[9.11, 3.23] [0.58, 1.29] [8.65, 9.98]	148	260
max_div_15_2.txt	15	2	4	49,82678080659435	[9.11, 3.23] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	605	1145
max_div_15_2.txt	15	2	5	79,12953032810968	[9.11, 3.23] [9.96, 8.17] [0.16, 4.62] [0.58, 1.29] [8.65, 9.98]	1487	2705
max_div_20_2.txt	20	2	2	8,510329018316506	[0.63, 5.96] [8.67, 3.17]	7	166
max_div_20_2.txt	20	2	3	21,99608586781776	[8.57, 8.36] [0.63, 5.96] [8.67, 3.17]	50	458
max_div_20_2.txt	20	2	4	40,00225775631908	[3.47, 9.43] [1.16, 4.47] [8.57, 8.36] [8.67, 3.17]	336	1754
max_div_20_2.txt	20	2	5	63,651679233818925	[3.47, 9.43] [8.57, 8.36] [1.97, 3.5] [0.63, 5.96] [8.67, 3.17]	2966	4775
max_div_30_2.txt	30	2	2	11,657143732492965	[9.84, 8.96] [2.07, 0.27]	18	141
max_div_30_2.txt	30	2	3	28,944299609596452	[1.91, 9.6] [9.84, 8.96] [2.07, 0.27]	42	221
max_div_30_2.txt	30	2	4	52,77117129353596	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [2.07, 0.27]	703	1866
max_div_30_2.txt	30	2	5	80,91024055749885	[1.91, 9.6] [9.84, 8.96] [8.0, 1.53] [0.65, 3.26] [2.07, 0.27]	28344	14751
max_div_15_3.txt	15	3	2	13,27323999632343	[0.3, 0.92, 4.23] [9.88, 9.88, 6.26]	2	43
max_div_15_3.txt	15	3	3	31,868525979035287	[1.71, 1.95, 9.22] [5.17, 1.39, 0.91] [9.88, 9.88, 6.26]	12	196
max_div_15_3.txt	15	3	4	59,76375886018705	[1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [6.65, 9.45, 1.02] [9.88, 9.88, 6.26]	85	717
max_div_15_3.txt	15	3	5	96,08583181098497	[1.15, 9.21, 3.11] [1.71, 1.95, 9.22] [0.3, 0.92, 4.23] [9.88, 9.88, 6.26] [9.47, 3.56, 1.83]	370	1720
max_div_20_3.txt	20	3	2	11,800309317979762	[9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	6	120
max_div_20_3.txt	20	3	3	30,872657951628632	[8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07]	44	374
max_div_20_3.txt	20	3	4	56,690310242464825	[0.83, 7.06, 3.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	298	1453
max_div_20_3.txt	20	3	5	92,8297431636503	[0.83, 7.06, 3.34] [8.87, 9.56, 5.34] [9.81, 2.05, 1.83] [0.65, 3.76, 9.07] [9.6, 2.02, 9.0]	1078	3241
max_div_30_3.txt	30	3	2	13,07373703269268	[6.71, 0.35, 9.42] [8.06, 9.59, 0.27]	26	213

max_div_30_3.txt	30	3	3	34,29052905319284	[8.32, 0.47, 8.98] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	143	445
max_div_30_3.txt	30	3	4	63,70195955868694	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	1072	2030
max_div_30_3.txt	30	3	5	99,59204059823645	[8.32, 0.47, 8.98] [6.75, 0.07, 1.8] [7.54, 8.93, 9.66] [8.06, 9.59, 0.27] [1.53, 8.09, 9.56]	36699	15822

A modo de conclusión, podríamos decir casi sin temor a equivocarnos que el algoritmo de ramificación y acotación que hemos implementado asegura encontrar una solución óptima al problema, siempre adaptando el coeficiente de desviación (véase el informe explicativo del algoritmo para más información sobre las cotas) al problema que estemos ejecutando.

Si nos fijamos, el resultado y su valor objetivo son constantes que se repiten para las distintas ejecuciones de un mismo problema, sin importar la estrategia de exploración del árbol que estemos utilizando o el algoritmo utilizado para generar la cota inicial inferior. Esto se debe a que el algoritmo encuentra la solución óptima en todos esos casos. Donde sí observamos diferencias es en el tiempo necesario para ejecutar el algoritmo, así como en el número de nodos que son expandidos en cada ejecución. El tiempo viene directamente relacionado con el número de nodos que se expanden, excepto en ciertos casos donde el tiempo medido no es demasiado coherente. Esto probablemente se deba a imprecisiones en el cálculo de los milisegundos por parte de Java.

El número de nodos expandidos mantiene cifras bastante similares en los tres primeros casos: aquellos en los que se utiliza la técnica de exploración en profundidad. Estas medidas, además, son bastante positivas e indican que la poda se lleva a cabo con éxito. Para un problema, por ejemplo, con 30 entradas y soluciones de tamaño `m = 3`, sabemos que habrá cuatro niveles en el árbol: 0, 1, 2 y 3. El número de nodos de cada uno de ellos, si se construyese el árbol completo, sin poda, sería de 30^n (siendo n el número del nivel). Esto daría un total de 27.931 nodos. En cambio, si observamos las tablas para estos casos, vemos que tan solo se han expandido 145 nodos.

En cambio, cuando pasamos a explorar el árbol expandiendo siempre el nodo con la cota superior más pequeña, evidentemente los resultados empeoran. Se sigue obteniendo la solución óptima global, pero el tiempo necesario aumenta de forma considerable (en el caso más extremo, llega a 37 segundos). Esto se relaciona con el incremento en el número de nodos generados. Para los ejemplos anteriores, pasa de 145 a 221 nodos generados. Para otros casos, empeora aún más y llega a generar 15.000 nodos para el último problema (30_3 con m = 5), frente a los 1.000 que generaba explorando en profundidad.

Todo esto se debe a que la exploración se lleva a cabo siempre a través del nodo con el peor potencial. La cota superior más pequeña, aunque no asegura que el nodo en cuestión conduzca a la peor solución de toda esa rama, sí nos lleva a la zona menos prometedora. Si, por el contrario, utilizásemos la cota superior más alta, probablemente los resultados mejorarían bastante, incluso superando en muchos casos a los obtenidos por la exploración en profundidad.