Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Informe de la Práctica 8 de DAA
Curso 2023-2024

Maximum Diversity Problem

Álvaro Fontenla León

La Laguna, 7 de mayo de 2024

Licencia

Resumen

Resolución del problema de diversidad máxima utilizando los algoritmos voraz, búsqueda local, GRASP, búsqueda tabú y ramificación y poda.

Índice general

Índice de Tablas

Capítulo 1

Introducción

1.1. Contexto

En esta práctica nos encontramos ante un problema de diversidad máxima. El principal objetivo de esta práctica es maximizar la distancia entre un conjunto de puntos.

1.1.1. Objetivos

Durante el desarrollo de esta práctica, se han cumplido los objetivos listados a continuación:

- Maximización de la diversidad.
- Desarrollo de diferentes algoritmos buscando una mayor optimalidad.
- Resolución de instancias de distintas características.

Capítulo 2

Maximum Diversity Problem

2.1. Descripción

En el Maximum diversity problem se desea encontrar el subconjunto de elementos de diversidad máxima de un conjunto dado de elementos. Sea dado un conjunto S = {s_1,s_2, . . . ,s_n} de n elementos, en el que cada elemento s_i es un vector s_i = $s_{i1}, s_{i2}, ..., s_{iK}$. Sea, asimismo, d_{ij} la distancia entre los elementos i y j. Si m <n es el tamaño del subconjunto que se busca el problema es descrito como: "Buscar los elementos que maximicen la distancia entre ellos". Para resolver este problema, la distancia utilizada será la euclídea.

Capítulo 3

Algoritmos

3.1. Algoritmo consturctivo voraz

Un constructivo voraz

Un algoritmo constructivo voraz para el Maximum diversity problem parte del subconjunto S formado por el elemento más alejado del centro de gravedad de S. A continuación, añade iterativamente a este subconjunto el elemento más alejado de su centro de gravedad hasta que S tenga m elementos

3.2. Búsqueda Local

Las búsquedas locales son un apartado importante en la resolución de este problema ya que nos permitirá la generación de soluciones de mayor calidad, por lo tanto, se han de escoger métodos de búsqueda local de calidad. Teniendo esto en cuenta, se ha desarrollado el método de intercambio de puntos entre los puntos de la solución y los puntos que no están incluidos en la misma.

3.3. GRASP

El algoritmo GRASP desarrollado se puede separar en dos partes: la fase constructiva y la fase de mejora.

3.3.1. Fase Constructiva

La fase constructiva de este algoritmo se ha desarrollado teniendo en cuenta la aleatoriedad necesaria y manteniendo un cierto nivel de optimalidad. Se ha utilizado el algoritmo voraz desarrollado en gran medida para esta fase.

En este caso, se construye la mitad de la solución con el algoritmo voraz, y la otra mitad se construye teniendo en cuenta la aleatoriedad, ordenando los puntos que más incrementan la dispersión y escogiendo uno de los tres primeros de manera aleatoria,

3.3.2. Fase de Mejora

Una vez construida una solución, se procede a su mejora, haciendo uso de la búsqueda local mencionada anteriormente.

Esta fase se ha implementado buscando la mayor optimalidad posible, ejecutando la fase de mejora múltiples veces, hasta que la mejor solución encontrada, no se pueda mejorar más.

3.4. Búsqueda Tabú

Este algoritmo se ha desarrollado haciendo que la perturbación de la solución sea el mismo método que la búsqueda local, es decir, el intercambio de puntos incluidos en la solución y los que no. Por otra parte, se ha utilizado una memoria a corto plazo, haciendo que los movimientos estén prohibidos por cinco iteraciones del algoritmo. El algoritmo desarrollado es multiarranque, esto es, se comienza con una solución factible haciendo uso de la fase constructiva del algoritmo GRASP. Luego, se realiza una búsqueda local, marcando como tabú ese movimiento, y verificando si es una mejor solución. Este proceso se repite un determinado número de veces. Cabe destacar que existe un criterio de parada, si la solución no ha mejorado en n iteraciones, se para la búsqueda y se comienza con una nueva solución generada por GRASP.

3.5. Ramificación y Poda

Este último algoritmo se ha desarrollado de manera constructiva, implementando un árbol de soluciones parciales, en el que, en cada nivel, se añade un elemento a la solución. Se toma como cota inferior la distancia que arroja el algoritmo voraz. Comenzando con una solución vacía, se crea un nodo por cada elemento distinto que se pueda incluir en la solución. Una vez generados los hijos del nodo, se calcula la nueva cota superior y se eliminan los nodos que no tienen un valor mayor que dicha cota. De manera recursiva para cada nodo del árbol, se ejecutan los procesos anteriores, almacenando la solución con mayor valor objetivo.

Capítulo 4

Experimentos y resultados computacionales

4.1. Constructivo voraz

Algoritmo voraz					
Problema	n	K	m	z	$CPU(ms)$
$max\_div\_15\_2$	15	2	2	11.8592	<1
$max\_div\_15\_2$	15	2	3	25.7262	<1
$max\_div\_15\_2$	15	2	4	48.4139	<1
$max\_div\_15\_2$	15	2	5	73.5619	<1
$max\_div\_20\_2$	20	2	2	8.51033	<1
$max\_div\_20\_2$	20	2	3	21.9961	<1
$max\_div\_20\_2$	20	2	4	39.5682	<1
$max\_div\_20\_2$	20	2	5	61.2393	<1
$max\_div\_30\_2$	30	2	2	11.6571	<1
$max\_div\_30\_2$	30	2	3	28.9443	<1
$max\_div\_30\_2$	30	2	4	52.7712	<1
$max\_div\_30\_2$	30	2	5	80.9102	<1
$max\_div\_15\_3$	15	3	2	13.2732	<1
$max\_div\_15\_3$	15	3	3	30.3241	<1
$max\_div\_15\_3$	15	3	4	59.7638	<1
$max\_div\_15\_3$	15	3	5	94.7487	<1
$max\_div\_20\_3$	20	3	2	11.8003	<1
$max\_div\_20\_3$	20	3	3	30.8727	<1
$max\_div\_20\_3$	20	3	4	56.5347	<1
$max\_div\_20\_3$	20	3	5	92.8297	<1
$max\_div\_30\_3$	30	3	2	13.0737	<1
$max\_div\_30\_3$	30	3	3	33.8423	<1
$max\_div\_30\_3$	30	3	4	63.5184	<1
$max\_div\_30\_3$	30	3	5	99.5088	<1

Cuadro 4.1: Algoritmo voraz. Tabla de resultados

4.2. Búsquda Local

Búsqueda Local

Problema	n	K	m	z	$CPU(ms)$
$max\_div\_15\_2$	15	2	2	11.8592	<1
$max\_div\_15\_2$	15	2	3	27.3727	<1
$max\_div\_15\_2$	15	2	4	49.5462	<1
$max\_div\_15\_2$	15	2	5	76.6535	<1
$max\_div\_20\_2$	20	2	2	8.51033	<1
$max\_div\_20\_2$	20	2	3	21.9961	<1
$max\_div\_20\_2$	20	2	4	40.0023	<1
$max\_div\_20\_2$	20	2	5	62.8729	<1
$max\_div\_30\_2$	30	2	2	11.6571	<1
$max\_div\_30\_2$	30	2	3	28.9443	<1
$max\_div\_30\_2$	30	2	4	52.7712	<1
$max\_div\_30\_2$	30	2	5	80.9102	<1
$max\_div\_15\_3$	15	3	2	13.2732	<1
$max\_div\_15\_3$	15	3	3	31.8685	<1
$max\_div\_15\_3$	15	3	4	59.7638	<1
$max\_div\_15\_3$	15	3	5	96.0858	<1
$max\_div\_20\_3$	20	3	2	11.8003	<1
$max\_div\_20\_3$	20	3	3	30.8727	<1
$max\_div\_20\_3$	20	3	4	56.6903	<1
$max\_div\_20\_3$	20	3	5	92.8297	<1
$max\_div\_30\_3$	30	3	2	13.0737	<1
$max\_div\_30\_3$	30	3	3	34.2905	<1
$max\_div\_30\_3$	30	3	4	63.702	<1
$max\_div\_30\_3$	30	3	5	99.592	1

Cuadro 4.2: Búsqueda Local. Tabla de resultados

4.3. GRASP

GRASP					
Problema	n	K	m	z	$CPU(ms)$
$max\_div\_15\_2$	15	2	2	11.8592	<1
$max\_div\_15\_2$	15	2	3	26.146	<1
$max\_div\_15\_2$	15	2	4	48.8422	<1
$max\_div\_15\_2$	15	2	5	78.1886	<1
$max\_div\_20\_2$	20	2	2	8.51033	<1
$max\_div\_20\_2$	20	2	3	21.4584	<1
$max\_div\_20\_2$	20	2	4	39.8292	<1
$max\_div\_20\_2$	20	2	5	61.2393	<1
$max\_div\_30\_2$	30	2	2	11.6571	<1
$max\_div\_30\_2$	30	2	3	26.9329	<1
$max\_div\_30\_2$	30	2	4	52.7712	<1
$max\_div\_30\_2$	30	2	5	79.0227	1
$max\_div\_15\_3$	15	3	2	13.2732	<1
$max\_div\_15\_3$	15	3	3	31.8685	<1
$max\_div\_15\_3$	15	3	4	59.7638	<1
$max\_div\_15\_3$	15	3	5	96.0858	<1
$max\_div\_20\_3$	20	3	2	11.8003	<1
$max\_div\_20\_3$	20	3	3	30.8727	<1
$max\_div\_20\_3$	20	3	4	56.6903	<1
$max\_div\_20\_3$	20	3	5	89.2197	<1
$max\_div\_30\_3$	30	3	2	13.0737	<1
$max\_div\_30\_3$	30	3	3	34.2905	<1
$max\_div\_30\_3$	30	3	4	63.5184	<1
$max\_div\_30\_3$	30	3	5	98.6471	1

Cuadro 4.3: GRASP. Tabla de resultados

4.4. Búsquda Tabú

Búsqueda Tabú

Problema	n	K	m	z	$CPU(ms)$
$max\_div\_15\_2$	15	2	2	11.8592	288
$max\_div\_15\_2$	15	2	3	27.3727	467
$max\_div\_15\_2$	15	2	4	49.8268	719
$max\_div\_15\_2$	15	2	5	79.1295	905
$max\_div\_20\_2$	20	2	2	8.51033	348
$max\_div\_20\_2$	20	2	3	21.9961	683
$max\_div\_20\_2$	20	2	4	40.0023	936
$max\_div\_20\_2$	20	2	5	63.6517	1347
$max\_div\_30\_2$	30	2	2	11.6571	544
$max\_div\_30\_2$	30	2	3	28.9443	1029
$max\_div\_30\_2$	30	2	4	52.7712	1699
$max\_div\_30\_2$	30	2	5	80.9102	2270
$max\_div\_15\_3$	15	3	2	13.2732	266
$max\_div\_15\_3$	15	3	3	31.8685	512
$max\_div\_15\_3$	15	3	4	59.7638	752
$max\_div\_15\_3$	15	3	5	96.0858	1002
$max\_div\_20\_3$	20	3	2	11.8003	358
$max\_div\_20\_3$	20	3	3	30.8727	701
$max\_div\_20\_3$	20	3	4	56.6903	1072
$max\_div\_20\_3$	20	3	5	92.8297	1641
$max\_div\_30\_3$	30	3	2	13.0737	592
$max\_div\_30\_3$	30	3	3	34.2905	1072
$max\_div\_30\_3$	30	3	4	63.702	1745
$max\_div\_30\_3$	30	3	5	99.592	2555

Cuadro 4.4: Búsqueda Tabú. Tabla de resultados

4.5. Ramificación y Poda

Ramificación y Poda

Problema	n	K	m	z	$CPU(ms)$
$max\_div\_15\_2$	15	2	2	11.8592	<1
$max\_div\_15\_2$	15	2	3	25.8634	<1
$max\_div\_15\_2$	15	2	4	49.8268	<1
$max\_div\_15\_2$	15	2	5	79.1295	<1
$max\_div\_20\_2$	20	2	2	8.51033	<1
$max\_div\_20\_2$	20	2	3	17.5475	<1
$max\_div\_20\_2$	20	2	4	35.0221	1
$max\_div\_20\_2$	20	2	5	57.0974	1
$max\_div\_30\_2$	30	2	2	11.6571	1
$max\_div\_30\_2$	30	2	3	23.9842	1
$max\_div\_30\_2$	30	2	4	46.6093	2
$max\_div\_30\_2$	30	2	5	79.6417	2
$max\_div\_15\_3$	15	3	2	13.2732	<1
$max\_div\_15\_3$	15	3	3	31.8685	<1
$max\_div\_15\_3$	15	3	4	59.7638	1
$max\_div\_15\_3$	15	3	5	96.0858	1
$max\_div\_20\_3$	20	3	2	11.8003	<1
$max\_div\_20\_3$	20	3	3	30.8727	<1
$max\_div\_20\_3$	20	3	4	56.6903	1
$max\_div\_20\_3$	20	3	5	92.8297	2
$max\_div\_30\_3$	30	3	2	13.0737	<1
$max\_div\_30\_3$	30	3	3	34.2905	1
$max\_div\_30\_3$	30	3	4	63.702	2
$max\_div\_30\_3$	30	3	5	99.592	2

Cuadro 4.5: Ramificación y Poda. Tabla de resultados

4.6. Análisis compartativo entre algoritmos

Como podemos observar en las tablas de resultados, hay bastantes diferencias entre los algoritmos. La primera diferencia notable es el tiempo que consumen los algoritmos. Observamos que el algoritmo de búsquda tabú tiene un consumo de tiempo mucho mayor que el resto.

Por otro lado, los resultados arrojados por el algoritmo de búsqueda tabú son de mayor calidad respecto a los otros algoritmos. Por lo tanto, aunque el algoritmo de búsqueda tabú consuma más tiempo que el resto, éste arrojará mejores resultados, aunque el algoritmo de búsqueda local se acerca bastante a esos resultados, con un tiempo de ejecución mucho menor.