Problema de la Mochila

Nombre: Angel Azahel Ramirez Cabello **Matricula: A01383328**

Nombre: Luis Angel López Chávez **Matricula: A01571000**

Nombre: Franco Mendoza Muraira **Matricula: A01383399**

Nombre: Matricula:

Instrucciones: *Resuelve completa y correctamente cada uno de los siguientes puntos. Se evalúa el procedimiento. Resultado sin procedimiento no tiene puntaje alguno.*

1. **Un armador tiene un carguero con capacidad de hasta 700 toneladas. El carguero transporta contenedores de diferentes pesos para una determinada ruta. En la ruta actual el carguero puede transportar algunos de los siguientes contenedores:**

Contenedor	$c1$	$c2$	$c3$	$c4$	$c5$	$c6$	$c7$	$c8$	$c9$	$c10$
Peso [ton]	100	155	50	112	70	80	60	118	110	55
Beneficio [$ /ton]	1741	1622	1016	1264	1305	1389	1797	1330	1559	1578

El analista de la empresa del armador desea determinar el envío (conjunto de contenedores) que maximiza el beneficio.

- Resuelve a optimalidad, usando GAMS.
- Recopila información sobre la función objetivo, el tiempo computacional y el número de iteraciones.

SOLUTION:
---
c1 --> c2 --> c3 --> c5 --> c6 --> c7 --> c9 --> c10

Función Objetivo: 12007
Costo Total: 680
Execution Time: 60 ms
Número de iteraciones: 2

Para la modelación matemática se intenta maximizar el beneficio de los contenedores incluidos en el carguero mientras se respete la restricción de capacidad de 700 toneladas.

- Elabora un algoritmo heurístico para obtener una solución factible al problema. Presenta un pseudocódigo o diagrama de flujo.

Pseudocódigo

Búsqueda Voraz (Greedy)

```
costo_actual = 0
beneficio_actual = 0
costo_maximo = 700

nodos = nodos_iniciales // Lista de nodos, cada uno con un beneficio y costo
nodos_no_visitados = nodos_iniciales // Copia de la lista de nodos iniciales

// Restricciones del problema
while (costo_actual < costo_maximo) and (nodos_no_visitados != ∅):
    // Filtrar nodos imposibles según la restricción de costo máximo
    nodos_posibles = obtener_nodos_posibles(nodos)

    // Si ningún nodo cumple con restricción de costo máximo
    if nodos_posibles == ∅:
        break

mejor_nodo = obtener_maximo_beneficio(nodos_posibles)

nodos_visitados.append(mejor_nodo) # Agregar el mejor nodo a la solución
nodos_no_visitados.remove(mejor_nodo) # Quitar nodo visitado del problema

costo_actual += mejor_nodo.costo
beneficio_actual += mejor_nodo.beneficio
```

- Implementa tu algoritmo heurístico en el ejemplo 1. Recopila información sobre la función objetivo y el tiempo computacional.

SOLUTION:

---

c7 --> c1 --> c2 --> c10 --> c9 --> c6 --> c8

Función Objetivo: 11016
Costo Total: 678

Execution Time: 0.798523999947065ms

2. **En el archivo de Excel se muestran varios casos que se le presentan al carguero.**
   - Resuelve cada uno a optimalidad, usando GAMS.
   - Utilizando el algoritmo heurístico utilizado anteriormente resuelva los nuevos casos presentados.
   - Realice una o varias tablas donde se recopila la información del valor objetivo, el tiempo de cómputo, así también el valor porcentual de la diferencia obtenida del valor objetivo entre el modelo matemático y el heurístico propuesto, realice lo mismo con el tiempo.

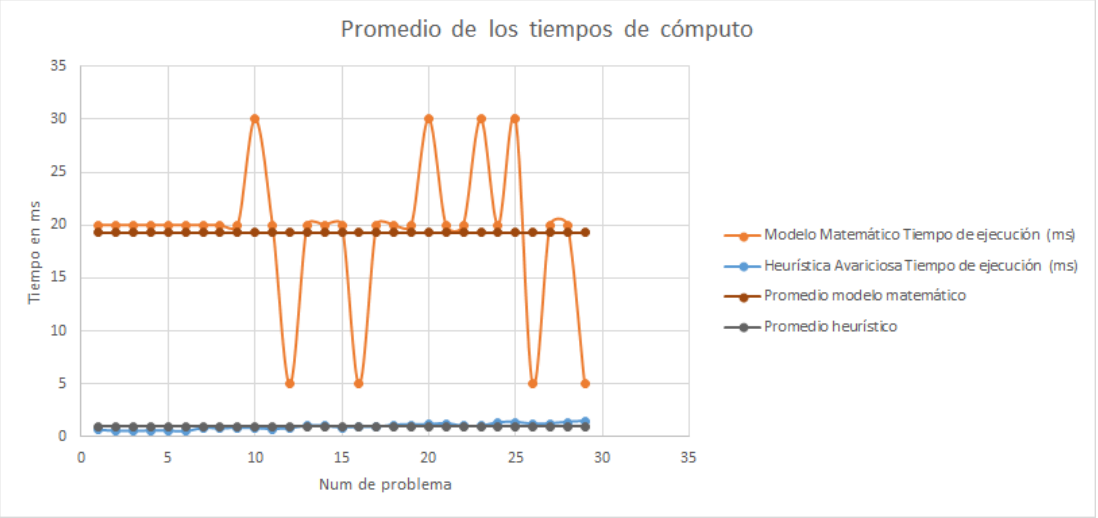
	Modelo Matemático			**Heurística Voraz**		Diferencia % F.O	Diferencia % Tiempo
	Función Objetivo (F.O)	Tiempo de ejecución (ms)	Número de Iteraciones	Función Objetivo (F.O)	Tiempo de ejecución (ms)	(Modelo - Heurística)	
Problema 1	11149	20	2	11149	0.64569899	0%	187%
Problema 2	13119	20	3	13119	0.59750799	0%	188%
Problema 3	11046	20	5	10315	0.56754399	7%	189%
Problema 4	12911	20	2	12053	0.60602899	7%	188%
Problema 5	11952	20	2	11952	0.59778700	0%	188%
Problema 6	12341	20	2	12341	0.54107400	0%	189%
Problema 7	11984	20	18	11900	0.84475200	1%	184%
Problema 8	13565	20	5	12416	0.80459200	9%	185%
Problema 9	13624	20	8	13088	0.87171099	4%	183%
Problema 10	14517	30	6	13856	0.83902500	5%	189%
Problema 11	12477	20	15	11997	0.74494599	4%	186%
Problema 12	13431	<10	3	13431	0.82142500	0%	144%
Problema 13	15503	20	1	14320	1.04101099	8%	180%
Problema 14	14153	20	9	14034	1.03521400	1%	180%
Problema 15	12551	20	2	11611	0.87722900	8%	183%
Problema 16	14464	<10	3	14464	0.95824699	0%	136%
Problema 17	14779	20	2	14260	0.92863399	4%	182%
Problema 18	12747	20	4	12747	1.12698800	0%	179%
Problema 19	15606	20	12	15074	1.13788400	3%	178%
Problema 20	13459	30	19	13256	1.15855699	2%	185%

	Modelo Matemático			Heurística Voraz		Diferencia % F.O	Diferencia % Tiempo
	Función Objetivo (F.O)	Tiempo de ejecución (ms)	Número de Iteraciones	Función Objetivo (F.O)	Tiempo de ejecución (ms)	(Modelo - Heurística)	
Problema 21	15025	20	7	13853	1.26681399	8%	176%
Problema 22	13463	20	10	11948	1.01342299	12%	181%
Problema 23	13387	30	11	12512	1.03172199	7%	187%
Problema 24	15368	20	4	14265	1.33372400	7%	175%
Problema 25	13090	30	27	12450	1.38645499	5%	182%
Problema 26	12917	<10	3	12281	1.2069580	5%	122%
Problema 27	13169	20	12	12542	1.2458599	5%	177%
Problema 28	14758	20	9	14104	1.3918329	5%	174%
Problema 29	15180	<10	1	14502	1.4660059	5%	109%

- Determine el promedio de los valores objetivo, del tiempo, y las diferencias porcentuales para ambos modelos.

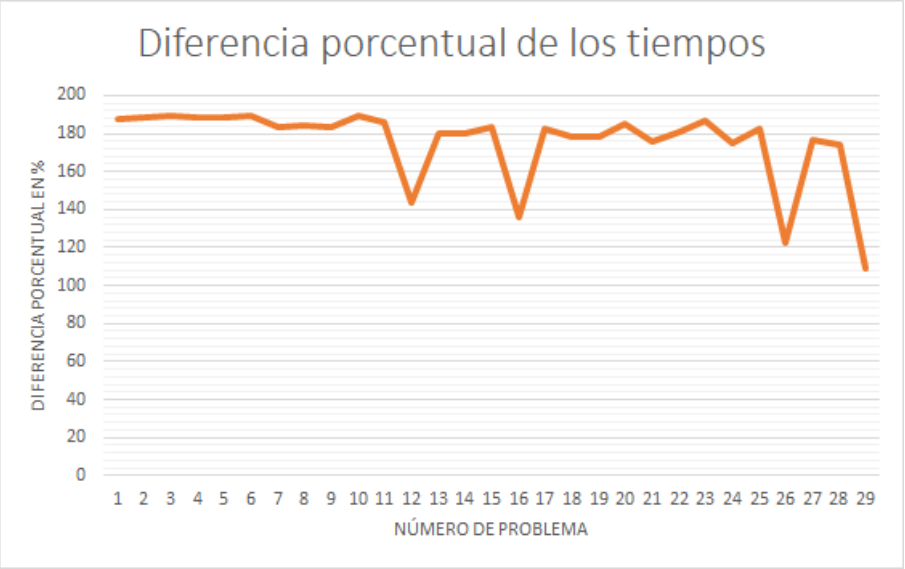
Modelo Matemático			Heurística Voraz		Diferencia % F.O	Diferencia % Tiempo
Promedio Función Objetivo	Promedio Tiempo de ejecución (ms)	Promedio de Número de Iteraciones	Promedio Función Objetivo	Promedio Tiempo de ejecución (ms)	(Modelo - Heurística)	
13508.1034	19.3103448	7.13793103	12960	0.96857416	4%	175%

- Grafique el promedio de los tiempos de cómputo, tanto para el modelo exacto y el heurístico. Explique



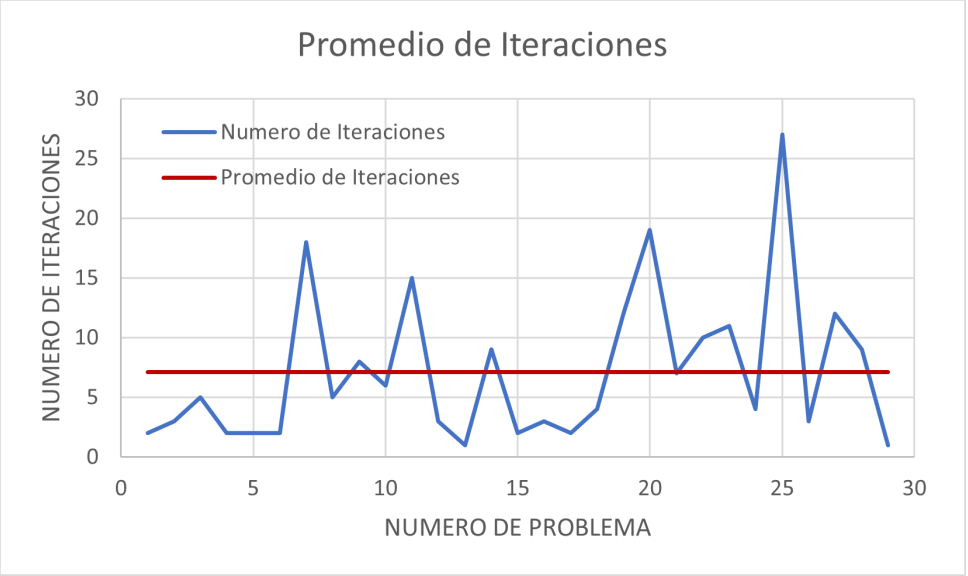
La diferencia existente entre ambas mediciones se debe principalmente a la búsqueda por la respuesta óptima por parte del código generado en GAMS que está basado en el modelo matemático y aunque no se trata de una diferencia en promedio muy abrumadora en la ejecución de los 29 problemas, al momento de visualizar un problema de más de 20 nodos quizá esta diferencia se vuelva crucial entre esperar 1 seg o 1 día para resolver el problema.

- Grafique la diferencia porcentual de los tiempos. Explique



La existencia de una diferencia porcentual mayor al 100% indica una distinción clave en la elección sobre cuál método utilizar para un problema grande, ya que, está magnitud de diferencia se mantiene constante en casi todos los problemas realizados, por lo que, será necesario voltear a ver al heurístico en búsqueda de un método para resolver un problema de más de 20 nodos.

- Grafique el número promedio de iteraciones realizadas por Gams. Explique



En el número de iteraciones realizadas por GAMS, se puede ver una gran variabilidad, llegando a iteraciones superando las 25, y otras quedándose por debajo de 5. A pesar de esto la cantidad de iteraciones se queda en un promedio de 7.13 lo cual es aceptable debido a la complejidad de algunos de los problemas, lo que también explica la variabilidad en la cantidad de iteraciones.

3. Conclusiones

En conclusión, pudimos encontrar un claro comportamiento al comparar los distintos métodos para encontrar una solución de qué contenedores poner en el carguero. Primeramente se pudo notar que el método del modelo matemático siempre obtuvo una función objetivo (maximizar beneficio) igual o mejor para todos los escenarios considerados, esto se debe ya que en el modelo matemático siempre se encuentra la solución óptima, aunque existen casos donde el método heurístico coincide en la misma solución óptima como se vio en 7 de los 30 escenarios probados.

Es por esto qué también se puede observar la eficiencia del método heurístico elegido, siendo búsqueda voraz, donde siempre se encontró mínimo una solución cercana a la del modelo matemático. Sin embargo, la real ventaja de este método se encuentra al comparar los tiempos de ambos modelos, donde el método de la heurística fue considerablemente más rápido que el primer modelo.

Con estas comparaciones se puede establecer que si se tiene como máxima prioridad encontrar la mejor solución a un problema de programación lineal, es mejor crear un modelo matemático determinista siempre y cuando sea posible. Por otro lado, un método heurístico como la búsqueda voraz es suficiente para poder aproximarse a la solución óptima, siempre tomando en cuenta que mientras el tamaño del problema crezca, más tardará el algoritmo, mas sí llegará mínimo a una solución factible.