Facultad de Ingeniería
Ingeniería Civil Informática
Optimización NRC:7061

Análisis del Knapsack Problem con Herramientas Tecnológicas

Tesis de pregrado para optar al título de Ingeniería Civil Informática

Felipe Pereira Alarcón

Profesor Guía: Dr. Gustavo Gatica

Providencia, Chile

2023

DEDICATORIA

Esta Tesis se dedica a mis padres, Manuel Pereira y Alejandra Alarcón, quienes me brindaron apoyo constante durante estos largos años de carrera. Incluso en momentos difíciles, como la llegada de la pandemia y mi propia depresión, ellos me brindaron fuerza, voluntad, garra y luz para seguir adelante. Admiro profundamente su dedicación y les agradezco el esfuerzo que realizaron, tanto en términos monetarios como emocionales y temporales, para que pudiera llegar a ser el Primer Profesional de la Familia Pereira Alarcón. Pero esto no termina aquí: mi camino académico se expandirá a futuro.

También se la dedico a mis hermanos, Agustín Pereira y Maximilian Pereira, quienes me brindaron su ayuda y apoyo constante en el proceso de manera anímica. En momentos de estrés, depresión, ansiedad y caídas, ellos estuvieron siempre a mi lado para ayudarme a levantarme. Desde el primer día, han creído en mi potencial como profesional y me han motivado a seguir adelante.

Sin el apoyo de mi familia, este logro no habría sido posible. Agradezco además a mis profesores y compañeros de carrera, quienes me brindaron la orientación y el apoyo necesarios para alcanzar mis metas. Les estaré eternamente agradecido por todo.

También quiero dedicar esta tesis a mis amigos cercanos, quienes me apoyaron en la mayoría de los momentos difíciles, acompañándome hasta altas horas de la noche para terminar trabajos sin dormir para ir a sus trabajos y estudios. Su amistad y apoyo incondicional significaron mucho para mí.

Quiero hacer una dedicación con mención especial al profesor Walter Uribe, quien me ayudó durante más de un año a adaptarme a la realidad universitaria. Es un hombre sabio, valiente y con gran experiencia, pero, sobre todo, un ser humano lleno de cariño. Me ayudó a llenar vacíos que tenía como estudiante y como persona, y gracias a sus constantes retos y correcciones, me convertí en uno de sus mejores alumnos. Profesor Walter, si llega a leer esto, quiero que sepa que lo aprecio mucho y que siempre tendrá un lugar especial en mi vida. Además, está cordialmente invitado a mi ceremonia de titulación.

Universidad Andrés Bello
Conectar · Innovar · Liderar

A mi profesor guía, el Doctor Gustavo Gatica, un excelente profesional y un muy buen profesor, quien me acompañó durante mis estudios y se ha convertido recientemente en un ejemplo a seguir gracias a sus estudios, investigaciones y presentaciones de libros. Le agradezco todo el apoyo y la confianza en mí y mis compañeros, incluso cuando teníamos dificultades con ciertas metodologías, y por ayudarme a seguir en el camino para convertirme en un profesional ejemplar.

Dedico esta tesis a mis amigos universitarios. A pesar de perder el contacto con algunos, otros se han alejado y algunos más no me han tratado de la mejor manera, agradezco los buenos momentos que compartimos. Recuerdo cuando les impartía clases hasta tarde en un box para luego buscar algún tipo de locomoción desde Providencia a Paine a altas horas de la noche, sabiendo que no había transporte disponible.

También agradezco su compañía en momentos de compras, estudio, juego y desestresamiento. Si bien he perdonado sus acciones negativas hacia mí, todavía me duele el mal trato que recibí, especialmente cuando siempre fui una persona tranquila y respetuosa.

A cada una de las personas con las que conviví estos años, les estoy profundamente agradecido, ya que, para mí, todas son importantes y todas han dejado una huella en mi vida.

AGRADECIMIENTOS

Agradezco sinceramente a la Facultad de Ingeniería Civil Informática de la Universidad Nacional Andrés Bello por brindarme una educación de excelencia, la cual me ha permitido desarrollar mis capacidades y enfrentar mis desafíos profesionales. Su sello de calidad es una verdadera visión de la ingeniería civil.

Quiero expresar mi gratitud a los profesores de mi comisión evaluadora por su disposición y comprensión con relación al espíritu de esta tesis. Su experiencia y sabiduría fueron fundamentales para guiar mi trabajo y lograr el éxito.

También deseo agradecer a todo el personal administrativo que trabajó detrás de escena para hacer posible mi desarrollo en la Universidad. En especial, mi agradecimiento al Doctor Gustavo Gatica por su paciencia y buena disposición en todo momento. Sus consejos y apoyo fueron de gran ayuda durante mi carrera universitaria.

Tabla de contenido

Tabla de Figuras

Universidad Andrés Bello
Conectar · Innovar · Liderar

Tablas

Universidad Andrés Bello
Conectar · Innovar · Liderar

CAPÍTULO 1

INTRODUCCIÓN

1.1 Transfondo

El problema de la mochila es uno de los problemas de optimización combinatoria más estudiados en la literatura. Es un problema clásico utilizado en diferentes áreas, como la logística, la informática, la economía y la ingeniería, entre otras [2]. El problema se centra en seleccionar un subconjunto de objetos con pesos y valores asociados de tal manera que maximice el valor total de los objetos llevados, sin exceder una capacidad máxima dada de la mochila [1].

Este problema, en sus diferentes variantes, genera un gran interés tanto en el ámbito teórico como práctico, siendo objeto de numerosos estudios. El interés práctico del problema se debe a situaciones clasificadas como un tipo de problema de la mochila tienen una gran cantidad de aplicaciones en la vida real, entre las cuales se pueden mencionar el presupuesto de inversiones, la carga de mercancías y el corte de productos [7].

Desde el punto de vista teórico, el problema de la mochila tiene una estructura simple que permite la aplicación de numerosas propiedades combinatorias. Además, otros problemas combinatorios más complejos pueden ser resueltos mediante una serie de subproblemas que caen dentro de la categoría del problema de la mochila. Esto explica el gran interés teórico que despiertan estos problemas [7].

1.2 Motivación

Este problema, conocido como "problema de la mochila", es un ejemplo de un problema **NP-COMPLETO** [1]. Esto significa que, aunque no se ha encontrado un algoritmo que resuelva en tiempo polinómico, se cree que no existe ninguna solución eficiente para

resolver el problema en todas las circunstancias. En otras palabras, es un problema muy difícil de resolver de manera óptima para grandes conjuntos de datos [7].

El problema de la mochila específicamente se refiere a la selección de objetos con un valor y un peso determinado, con el objetivo de llenar una mochila con una capacidad limitada de forma tal que maximice el valor total de los objetos seleccionados. Este problema presenta un gran interés práctico, ya que tiene aplicaciones en la planificación de producción, la distribución de recursos, la logística y el diseño de sistemas de transporte, entre otros [7].

1.3 Objetivos

El objetivo principal de este proyecto es aplicar algoritmos para resolver el problema de la mochila aplicando todos los recursos computacionales disponibles.

La intención es, demostrar que, con estos algoritmos, se puede lograr alcanzar el rendimiento óptimo para la solución objetiva.

Para lograr este objetivo, se estudiará el problema de la mochila, su historia, características y las diferentes variantes existentes. Luego, implementaremos cada algoritmo para resolver el problema, el cual servirá como base para el resto del informe.

Seguidamente, se aborda el problema de la mochila desde una perspectiva matemática y computacional. Se presenta el modelo matemático del problema, se describe su implementación utilizando AMPL y se realizan experimentos computacionales con Implementación en Python para validar la solución óptima obtenida.

Finalmente, se discuten algunas conclusiones relevantes y se proporcionan referencias bibliográficas para profundizar en el tema.

1.4 Estructura del documento

La tesis está estructurada de la siguiente manera:

- Prólogo: Sección conformada por una dedicatoria extensa dedicada a estos largos años de estudio, dedicados a mis cercanos y a mi Profesor Guía, y los agradecimientos a la Facultad.
- Introducción: Sección conformada por el trasfondo histórico y generalizado del Knapsack Problem, las motivaciones que llevan a buscar una solución óptima y objetivos a conseguir realizando el análisis de este.
- Explicación: Sección conformada por un caso introductorio considerando el escenario general para entenderlo, junto a un apto para que cualquier persona que lea este documento, tenga una facilidad para entender el desarrollo y contexto de este problema, acompañado de la descripción de la complejidad computacional de este problema.
- Modelo matemático: Sección importante, ya que contiene la definición formal de un modelo matemático para el análisis del problema de la mochila que utilizó Pisinger [1].
- Implementación en AMPL: Sección va de la mano con el punto anterior, ya que construye en un lenguaje de programación matemático como lo es AMPL, para su construcción con el modelo realizado antes, explicando brevemente la definición de conjuntos y parámetros, sumándole el objetivo del modelo a través de la función objetivo, respetando restricciones correspondientes para llevar todo esto al código y explicarlo paso a paso, para posteriormente, utilizar el ejemplo introductorio, y demostrar el correcto desarrollo de la implementación del modelo a código.
- Experimentos Computacionales: Sección encargada de introducir el caso de estudio utilizando tres métodos aplicados en algoritmos, comparando el tiempo de ejecución respecto a la capacidad de la instancia de la mochila, agregándole la modelación de análisis de datos TDA para cada método, luego agregando todo en Lenguaje Julia para su posterior análisis de datos amplios.
  Siguiendo, queda demostrado en subsecciones el análisis para cada método en secciones con gráficas para su respectivo análisis y finalizar con un resumen completo de toda la experimentación realizada con tablas que muestran la

información de cada instancia, la capacidad de cada instancia y los tiempos de ejecución secuencial y paralela.

- Problema planteado en Caso Nacional: Esta sección explica brevemente como este problema puede ser implementado en una empresa a nivel nacional como lo es Correos de Chile.
- Conclusiones: Esta sección da una finalización luego de una ardua investigación y análisis del Knapsack Problem.
- Referencias: Esta sección contiene cada una de las referencias ocupadas para este caso citadas en APA con su respectivo enlace para acceder.

CAPÍTULO 2

EXPLICACIÓN DEL PROBLEMA DE LA MOCHILA

2.1 Explicación Introductoria

El problema de la mochila es un problema matemático descrito de manera sencilla. Considere un escenario en el cual se dispone de una mochila con una capacidad máxima de peso que puedes cargar. También tienes una lista de objetos con diferentes pesos y valores. La pregunta es: ¿cómo elijo los objetos para poner en la mochila de manera que maximices el valor total, sin exceder la capacidad de peso máximo de la mochila?

2.2 Ejemplo Inductivo

Considere tener una mochila que puede cargar un máximo de 10 kilogramos, y tiene 5 objetos diferentes para elegir: una botella de agua que pesa 1 kg y tiene un valor de 5 dólares, un libro que pesa 3 kg y tiene un valor de 8 dólares, un par de zapatos que pesan 2 kg y tienen un valor de 10 dólares, una chaqueta que pesa 4 kg y tiene un valor de 15 dólares, y una cámara que pesa 5 kg y tiene un valor de 20 dólares.

Lista de Objetos		
Objetos	Peso(Kg)	Valor(USD)
Botella de agua	1	5
Libro	3	8
Par de Zapatos	2	10
Chaqueta	4	15
Cámara	5	20

Figura 1: Lista objetos

Universidad Andrés Bello
Conectar · Innovar · Liderar

Si solo pudieras llevar una parte de estos objetos en tu mochila, ¿cuáles deberías elegir para maximizar el valor total sin exceder el límite de peso de la mochila? Esta es la pregunta que el problema de la mochila intenta responder. [2]

En la práctica, este problema se aplica en situaciones como la planificación de rutas de reparto de paquetes, la optimización de la carga de aviones o barcos, o incluso en la selección de cartera de inversión para maximizar el rendimiento y minimizar el riesgo. El problema de la mochila tiene aplicaciones en muchos campos diferentes y se puede resolver con la ayuda de la programación matemática. [2]

2.3 Complejidad Computacional

Cualquier problema que se relacione con el problema de la mochila serán considerados **NP-COMPLETO**, lo cual ha sido demostrado por varios autores en la literatura. Esto significa que no existe un algoritmo cuya complejidad sea polinomial en función de *n*, a menos que se demuestre que **P = NP** en algún momento. [7]

No obstante, para el problema de la mochila única (que consta de un solo contenedor) existen algoritmos cuya complejidad temporal y espacial es pseudo-polinomial, es decir, está acotada por un polinomio en función de *n* y *c*, por ejemplo *O(nc)*. Sin embargo, esta complejidad no contradice el hecho que categorizar este problema como **NP-COMPLETO**, ya que un algoritmo tiene una complejidad pseudo-polinomial si depende de la longitud de la entrada (es decir, el número de bits requeridos para representarla) y el valor numérico de la entrada. Por lo general, la longitud de la entrada es exponencial en relación a su valor numérico. [7]

En otras palabras, el problema de la mochila única es débilmente **NP-COMPLETO**. Un problema es débilmente **NP-COMPLETO** si existe un algoritmo capaz de resolver el problema en tiempo polinomial en función de la dimensión del problema y la magnitud de los datos, en lugar del logaritmo en base 2 de sus magnitudes. Por lo tanto, estos algoritmos son técnicamente funciones exponenciales en relación al tamaño de la entrada y no son considerados algoritmos polinomiales. [7]

Por otro lado, a diferencia del problema de la mochila única, el problema de la mochila múltiple no es débilmente **NP-COMPLETO**, sino que es fuertemente **NP-COMPLETO**, significando que no puede existir un algoritmo capaz de resolver estos problemas en tiempo pseudo-polinomial a menos que evidencie **P = NP**. [7]

CAPÍTULO 3

MODELO MATEMÁTICO DEL PROBLEMA

3.1 Definición Formal

El problema de la mochila se puede modelar matemáticamente basándose en la propuesta de Pisinger [1] para la siguiente fórmula:

Maximizar Z:

$$\sum_{i=1}^{n} x_i \, v_i \tag{1}$$

Sujeto a:

$$\sum_{i=1}^{n} W_i \, x_i \leq W \tag{2}$$

Donde x_i en {0, 1} para todo *i* hasta *W*, además, se tienen los siguientes supuestos:

- ***n*** es el número de objetos. En AMPL se verá como **ELEMENTOS**.
- $\boldsymbol{V_i}$ es el valor del objeto *i*. En AMPL se verá como **VALOR.**
- $\boldsymbol{W_i}$ es el peso del objeto *i*. En AMPL se verá como **PESO**.
- ***W*** es la capacidad máxima de la mochila. En AMPL se verá como **PESOMAX.**
- $\boldsymbol{X_i}$ es una variable binaria que indica si se selecciona o no el objeto *i*. En AMPL se verá como **take{ELEMENTOS}.**

La idea es maximizar el valor total de los objetos seleccionados, sujeto a la restricción del peso total y que no exceda la capacidad máxima de la mochila. Además, se debe asegurar que cada objeto sea seleccionado completo (variable binaria $\boldsymbol{X_i}$ igual a 1) o no seleccionado (variable binaria $\boldsymbol{X_i}$ igual a 0).

La correlación entre el modelo matemático y los datos radica en cómo se utilizan los valores y pesos de los elementos, así como la capacidad de la mochila, para determinar la solución óptima. El modelo establece una relación entre las variables de decisión (si se selecciona o no cada elemento) y el valor total obtenido. A medida que se cambian los valores y pesos de los elementos, así como la capacidad de la mochila, el modelo matemático proporciona una solución óptima que maximiza el valor total.

3.2 Generalización del Modelo

El problema clásico de la mochila implica seleccionar un subconjunto de elementos para maximizar el valor total sin exceder la capacidad de la mochila. Sin embargo, en ciertos escenarios, es necesario considerar restricciones adicionales, objetivos alternativos o variantes del problema básico.

Restricciones adicionales

Para casos específicos, pueden existir restricciones adicionales que limiten la cantidad de elementos seleccionados o restrinjan los recursos disponibles. Estas restricciones pueden incluir límites en la cantidad de elementos seleccionados, restricciones de recursos adicionales como el tiempo o el espacio, o restricciones específicas del dominio en el que se aplica el problema. Por ejemplo, en un problema de mochila de viaje, se podría tener una restricción de tiempo que limite la duración total de los elementos seleccionados.

Objetivos alternativos

Además de maximizar el valor total, es posible que existan objetivos alternativos en el problema de la mochila. Estos objetivos pueden incluir la minimización del peso total en lugar de maximizar el valor, buscar un equilibrio óptimo entre el valor y el peso de los elementos seleccionados o incluso maximizar una función de utilidad ponderada que combine múltiples criterios. Por ejemplo, en un problema de mochila para una expedición al aire libre, se puede buscar maximizar la utilidad total considerando tanto el valor de los elementos como su peso, teniendo en cuenta la importancia relativa de cada uno.

Elementos fraccionales

En el problema clásico de la mochila, se asume que los elementos deben seleccionarse de forma discreta, es decir, se seleccionan por completo o no se seleccionan en absoluto. Sin embargo, en ciertos casos, es posible permitir la selección de fracciones de elementos, lo que se conoce como el problema de la mochila fraccional. Esta generalización permite una mayor flexibilidad en la selección de elementos y puede tener implicaciones en el valor y el peso total obtenido. La solución óptima para el

problema fraccional puede requerir la utilización de técnicas diferentes a las utilizadas en el problema discreto, como algoritmos de programación lineal.

Elementos múltiples

En la versión clásica del problema de la mochila, se asume que solo hay un elemento de cada tipo disponible para su selección. Sin embargo, en ciertos escenarios, puede ser necesario permitir múltiples copias de los elementos, lo que se conoce como el problema de la mochila múltiple. Esta generalización amplía las posibilidades de selección y puede influir en la complejidad y enfoques de resolución. En el problema de la mochila múltiple, se deben considerar tanto la cantidad de elementos seleccionados como las restricciones de capacidad. Esto puede requerir adaptaciones en los algoritmos utilizados, como la utilización de técnicas de programación dinámica o algoritmos de ramificación y poda.

Estas generalizaciones del problema de la mochila permiten adaptar el modelo básico a situaciones más específicas y complejas. Cada generalización introduce nuevos desafíos.

CAPÍTULO 4

IMPLEMENTACIÓN DEL MODELO EN AMPL

4.1 Definición del algoritmo

En esta implementación, se definen los siguientes conjuntos y parámetros:

- **ELEMENTOS**: conjunto que indica el rango de índices para los objetos.
- **PESOMAX**: parámetro que indica el peso máximo permitido en la mochila.
- **VALOR:** parámetro que indica el valor de cada objeto.
- **PESO:** parámetro que indica el peso de cada objeto.

Además, se definen dos parámetros adicionales:

- **acum_peso:** parámetro acumulativo para el peso de los objetos seleccionados.
- **acum_valor:** parámetro acumulativo para el valor de los objetos seleccionados.

Por último, se define la variable de decisión take, que es binaria y toma el valor 1 si se selecciona el objeto *i* y 0 en caso contrario.

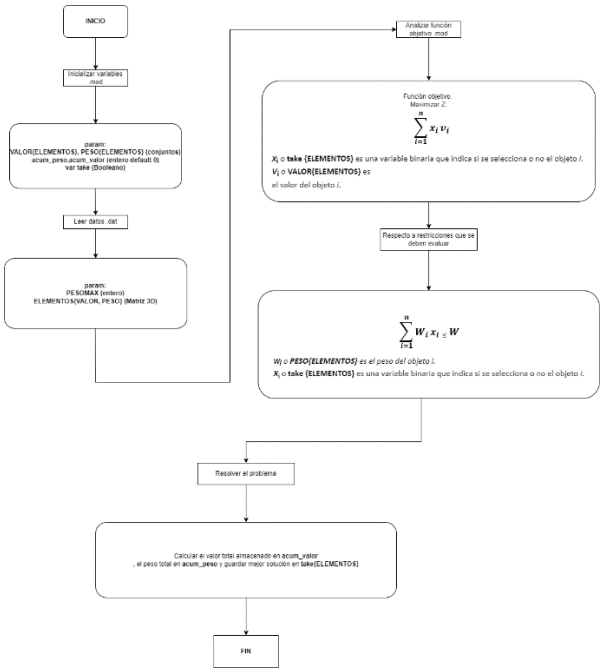

Figura 2: *Diagrama modelo AMPL*

El objetivo de la función objetivo es maximizar el valor total de los objetos seleccionados, el cual se calcula como la suma del valor de cada objeto multiplicado por la variable binaria take correspondiente.

De acuerdo con la restricción de peso, se debe respetar la suma de los pesos de los objetos seleccionados no debe exceder el peso máximo permitido en la mochila.

En resumen, este código representa una implementación del modelo matemático del problema de la mochila en AMPL, el cual es posible resolver mediante el uso de un solver de programación lineal entera para obtener la solución óptima del problema [1].

.

A continuación, se verá implementada el código en software AMPL y el paso a paso de su ejecución.

4.2 Implementación en Código AMPL

Código "knapsack.mod"

```
# Conjunto que indica el rango de índices para cada objeto
set ELEMENTOS;

# Peso mayor Permitido
param PESOMAX integer, >= 1
# Valor de cada objeto
param VALOR{ELEMENTOS} >= 0;
# Peso de cada objeto
param PESO{ELEMENTOS} >= 0;

# Acumula el peso de los objetos seleccionados
param acum_peso default 0;
# Acumula el valor de los objetos seleccionados
param acum_valor default 0;

# Binario, 1 Si tomamos el ítem i y 0 si no
var take{ELEMENTOS} binary;
# Función Objetivo que maximiza el Valor Total
maximize MaxVal:
        sum {i in ELEMENTOS} VALOR[i] * take[i];

# Restricción de Peso
subject to Weight:
        sum{i in ELEMENTOS} PESO[i] * take[i] <= PESOMAX;;
```

Código "knapsack.dat"

```
param PESOMAX := 102;

# Definimos ítems con valor y peso respectivamente
param: ELEMENTOS: VALOR PESO :=
  camara  15  2
  bolsa 100 20
  Mapa 90 20
  lapiz 60 30
  pistola 40 40
  azucar 15 30
  platano 10 60
  queso 1 10
  manteca 12 21
  chocolate 12 12
  sal 100  2
;
```


Universidad Andrés Bello
Conectar · Innovar · Liderar

4.3 Paso a Paso

Paso a Paso con Software AMPL:

Primero, guardamos el código como “knapsack.mod” y “knapsack.dat”

Luego, modelamos el problema luego de programarlo en “knapsack.mod” y le ingresamos la información del data “knapsack.dat”:

- model “ubicacionknapsack.mod”
- data “ubicacionknapsack.dat”

Utilizamos el Solver Simplex en:

- option solver cplex;

Y luego Resolvemos con:

- solve;

Para ver cuántos elementos eligió para la solución óptima se escribe en consola como:

- display take;

Resultado en pantalla:

```
take [*] :=
     Mapa  1
   azucar  0
    bolsa  1
   camara  1
chocolate  1
    lapiz  1
  manteca  0
  pistola  0
  platano  0
    queso  1
      sal  1
;
```

y para ver el valor y el peso totales que obtuvo de la solución optima se escribe en consola como:

- `display PESOMAX, sum{i in ELEMENTOS} VALOR[i] * take[i], sum{i in ELEMENTOS} PESO[i] * take[i];`

Universidad Andrés Bello
Conectar · Innovar · Liderar

Resultados demostrados en consola:

```
PESOMAX = 102
sum{i in ELEMENTOS} VALOR[i]*take[i] = 378
sum{i in ELEMENTOS} PESO[i]*take[i] = 96
```

4.4 Ejemplo Introductorio demostrado

¿Qué pasa si lo hacemos con el ejemplo que se explicó en Explicación del problema?

Para resolver este caso, Creamos un nuevo archivo .dat llamado “knapsack2.dat” y ingresamos los valores de la siguiente manera:

```
param PESOMAX := 10;

# Definimos ítems con valor y peso respectivamente
param: ELEMENTOS: VALOR PESO :=
  botella_agua 5 1
  libro 8 3
  par_zapatos 10 2
  chaqueta 15 4
  camara 20 5
;
```

Luego, ejecutamos el código con los siguientes comandos:

Donde modelamos el problema luego de programarlo en “knapsack.mod” y le ingresamos la información del data “knapsack2.dat”

- model “ubicacionknapsack.mod”
- data “ubicacionknapsack2.mod”

Utilizamos el Solver Simplex en:

- option solver cplex;

Y luego Resolvemos con:

- solve;

Para ver cuántos elementos eligió para la solución óptima se escribe en consola como:

- display take;

Resultado en pantalla:

```
take [*] :=
botella_agua  1
      camara  1
    chaqueta  1
       libro  0
 par_zapatos  0
   ;
```

Para ver el valor total y el peso total que obtuvo de la solución óptima se escribe en consola como:

- `display PESOMAX, sum{i in ELEMENTOS} VALOR[i] * take[i], sum{i in ELEMENTOS} PESO[i] * take[i];`

Resultados demostrados en consola:

```
PESOMAX = 10
sum{i in ELEMENTOS} VALOR[i]*take[i] = 40
sum{i in ELEMENTOS} PESO[i]*take[i] = 10
```

CAPÍTULO 5

EXPERIMENTOS COMPUTACIONALES

5.1 Caso de Estudio

El problema de la mochila es un problema de optimización combinatoria estudiado ampliamente y generó una gran cantidad de algoritmos y técnicas de resolución. Para evaluar el rendimiento de estos algoritmos, se han realizado experimentos computacionales utilizando diversas técnicas y conjuntos de datos.

Los experimentos computacionales suelen implicar la comparación del tiempo de ejecución y la calidad de la solución para cada algoritmo. Para realizar estas comparaciones, se utilizan conjuntos de datos que reflejan diferentes características y propiedades del problema de la mochila, como la distribución de los pesos y beneficios, la correlación entre ellos, la presencia de valores atípicos, etc.

Para realizar estos experimentos, se utilizarán técnicas comunes utilizadas en estos experimentos que incluyen la programación dinámica, Fuerza bruta y el método “greedy”, entre otras [5]. Además, se han propuesto algoritmos específicos para diferentes variantes del problema de la mochila, como el problema de la mochila en ejecución secuencial y paralela.

En consecuencia, se utilizará el Modelado TDA en cada caso, ya que esta técnica representa una rama emergente de las matemáticas aplicadas que se enfoca en el estudio de formas y patrones en datos mediante el uso de herramientas de topología algebraica. Se considera que el Modelado TDA es una herramienta poderosa para analizar y comprender la estructura de datos complejos.

Por otra parte, el Modelado TDA tiene como objetivo extraer información topológica de los datos, lo que permite identificar estructuras subyacentes que no son evidentes a simple vista. Para lograr esto, se utilizan técnicas de filtrado de datos y diagramas de persistencia, los cuales representan la evolución de los componentes topológicos en diferentes escalas.

En resumen, el Modelado TDA es una técnica valiosa para analizar datos en profundidad y obtener información valiosa que no es evidente a simple vista.

De igual manera, se utilizará el lenguaje de programación Julia para llevar a cabo la modelación y experimentación con pesos grandes.

Julia es un lenguaje de programación de alto rendimiento diseñado específicamente para computación científica y numérica. Una de sus principales ventajas es su capacidad para manejar de manera eficiente grandes conjuntos de datos y cálculos complejos, lo que lo hace especialmente adecuado para la experimentación.

En la práctica, la utilización de Julia permitirá una mayor eficiencia y rapidez en la experimentación, lo que a su vez facilitará la exploración de diferentes escenarios y opciones de modelación.

Las pruebas se realizaron en Jupyter Notebook desde Google Collab. Para acceder a este, debes tener una cuenta Google para ingresar.

toda esta información esta demostrada en Google Collab, QR disponible

5.2 Relaciones básicas de la estructura de datos

La relación entre el valor total de los elementos y la capacidad de la mochila es mostrada gráficamente como se expone en la Figura 5.2:

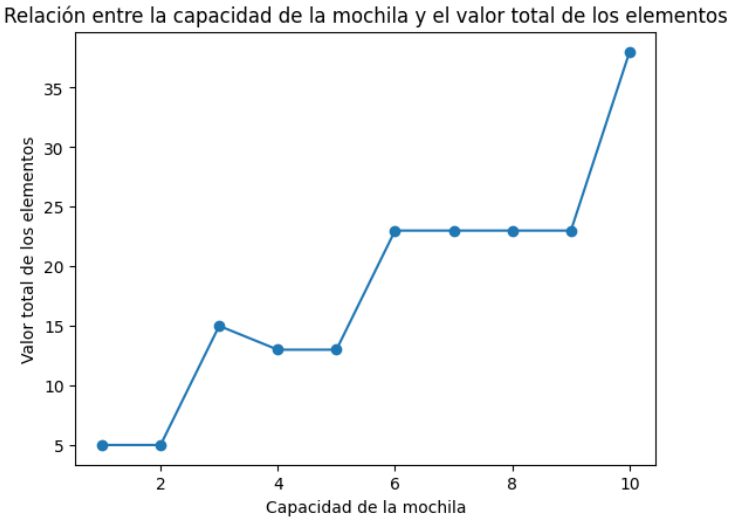

Figura 5.2: *Relación entre la capacidad de la mochila y el valor total de los elementos*

Valores = [5, 8, 10, 15, 20]; pesos = [1, 3, 2, 4, 5]

La implementación de un Algoritmo para encontrar la solución óptima siempre ha sido un problema, pero, que con los años se han encontrado múltiples soluciones aplicadas a un algoritmo. La forma más clara para ver esto es relacionar el tiempo respecto a la capacidad total que tiene la mochila en su respectivo caso.

Para analizar esta información, se desarrolló en código, las técnicas e implementación de su solución en Python.

5.3 Método Programación dinámica en Python

El método que se detalla a continuación es la “Programación dinámica”, el cual se encuentra descrito en Google Colab.

En este método, el problema se divide en subproblemas más pequeños y manejables, y se resuelven estos subproblemas en orden, guardando la solución de cada subproblema para utilizarla en subproblemas posteriores. En el caso del problema de la mochila, la programación dinámica utiliza una tabla o matriz para almacenar la solución óptima para cada subconjunto de elementos y capacidad de la mochila, y se va construyendo la solución para conjuntos mayores de elementos a partir de las soluciones óptimas de conjuntos más pequeños.

Las librerías usadas para realizar este método en Python son “Matplotlib” para la creación del gráfico de comparación del y la librería “time” para calcular los tiempos de inicio y final, para su respectivo uso.

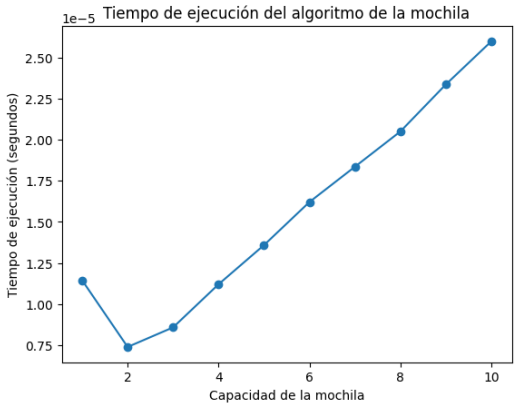

Figura 5.3: *Tiempo de ejecución respecto a la capacidad de la mochila.*

Paralelización.

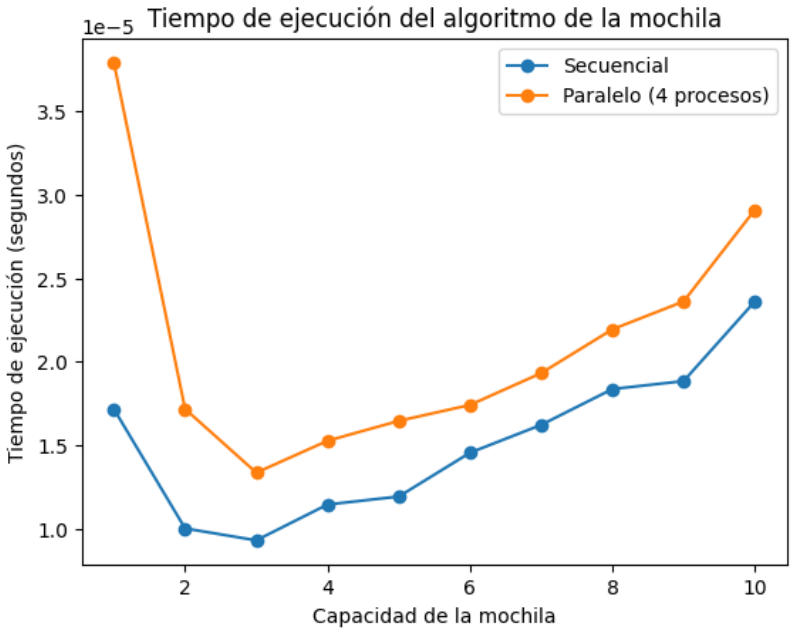

Figura 5.3.2: *Tiempo de ejecución respecto a la capacidad de la mochila. (Paralelo vs secuencial)*

Cuando utilizamos la biblioteca de multiprocessing para crear código paralelo, podemos ejecutar varias instancias de nuestro algoritmo de manera simultánea en diferentes procesadores, haciendo que el proceso sea más rápido que el código secuencial. En el código secuencial, sólo se ejecuta una instancia del algoritmo a la vez, mientras que la version paralela ejecuta varias instancias al mismo tiempo en diferentes procesadores.

En este caso específico, el algoritmo de la mochila es un problema factiblr para paralelizar de manera eficiente, ya que cada capacidad en la iteración es independiente de las demás. Por lo tanto, distribuye la carga de trabajo entre los diferentes procesos para acelerar el proceso.

Pero, el código secuencial es más rápido que el código paralelo en términos de velocidad de ejecución, ya que la estructura de datos ocupada es de una cantidad pequeña y limitada, por lo tanto, se deduce que para este caso, la secuencialidad es más eficiente.

Se presentará a continuación el modelado topológico de datos (TDA)

Aplicando el modelado TDA, el análisis muestra la siguiente resolución:

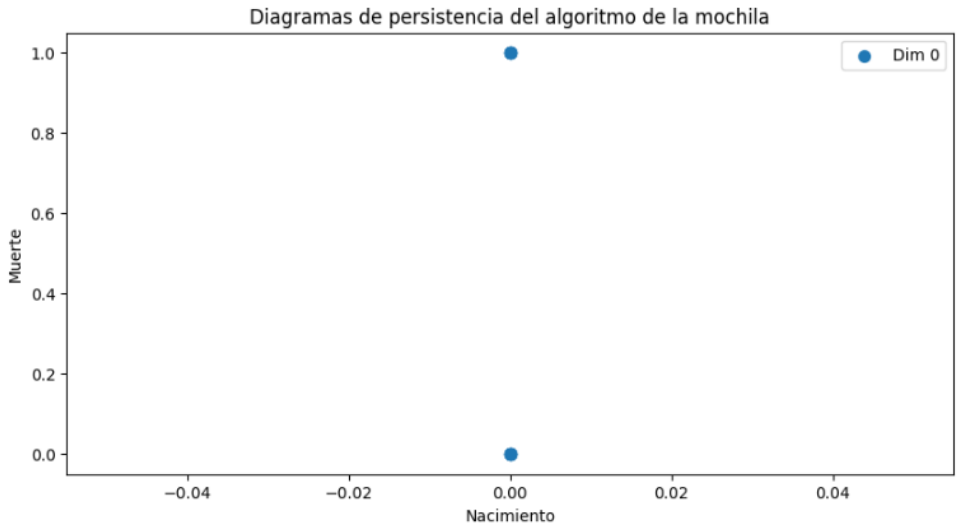

Figura 5.3.3: *Diagrama de persistencia algoritmo 2D*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 1 dimension), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escalas

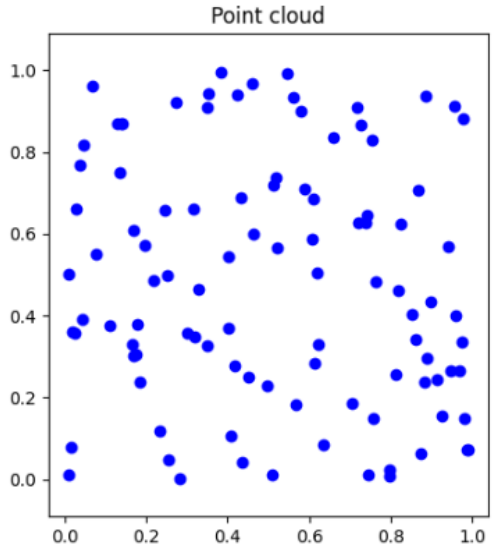

Figura 5.3.4: *Nube de Puntos*

En cuanto a las nubes de puntos, estas representan los datos en sí mismos. Cada punto en la nube representa un elemento de los datos y su posición en el espacio corresponde a sus características de peso o atributos.

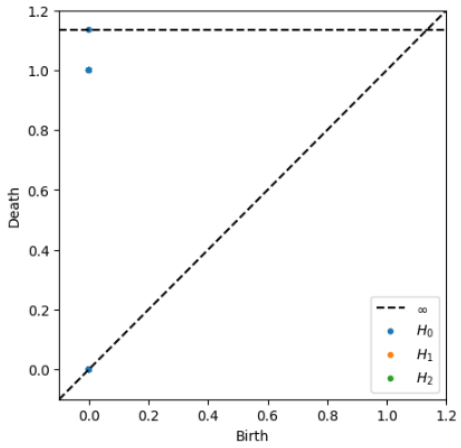

Figura 5.3.5: *Diagrama de Persistencia 2D*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 2 dimensiones), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escalas.

5.4 Método Greedy en Python

El método que se detalla a continuación es el “Greedy”, el cual se encuentra descrito en Google Colab.

En términos sencillos, el método greedy es una estrategia de solución que busca la mejor opción en cada paso, sin considerar cómo afectará esta decisión en el futuro. En el problema de la mochila, esto significa que el algoritmo selecciona los elementos más valiosos (en términos de su valor unitario) y los coloca en la mochila, sin considerar si el espacio restante de la mochila es suficiente para los elementos restantes o no. Por lo

tanto, el método greedy no siempre produce la solución óptima para el problema de la mochila, aunque a veces puede ser una solución rápida y efectiva.

Las librerías usadas para realizar este método en Python son “Matplotlib” para la creación del gráfico de comparación del y la librería “time” para calcular los tiempos de inicio y final, para su respectivo uso.

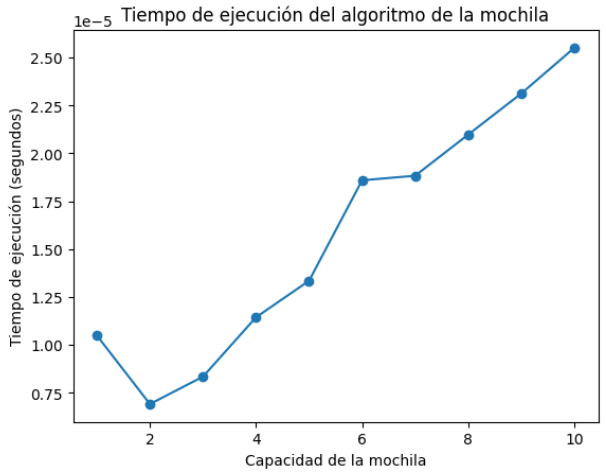

Figura 5.4: *Tiempo de ejecución respecto a la capacidad de la mochila*.

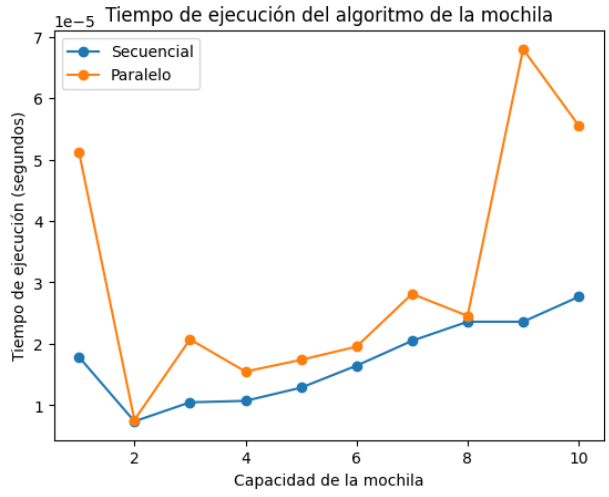

Figura 5.4.2: *Tiempo de ejecución respecto a la capacidad de la mochila.*

Para lograr la paralelización, se utiliza la librería de “Multiprocessing” para su posterior evaluación.

En este caso,el código secuencial es más rápido que el código paralelo en términos de velocidad de ejecución, ya que la estructura de datos que se está ocupando es de una cantidad pequeña y limitada. Pero, al ser un algoritmo que no siempre produce la solución óptima para el problema de la mochila, puede obtener tiempos similares a la secuencialidad respecto a la capacidad de la mochila hablando del tiempo de ejecución.

Se presentará a continuación el modelado topológico de datos (TDA)

Aplicando el modelado TDA, el análisis muestra la siguiente resolución:

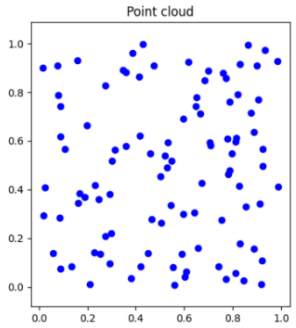

Figura 5.4.3: *Nube de Puntos*

En cuanto a las nubes de puntos, estas representan los datos en sí mismos. Cada punto en la nube representa un elemento de los datos y su posición en el espacio corresponde a sus características de peso o atributos.

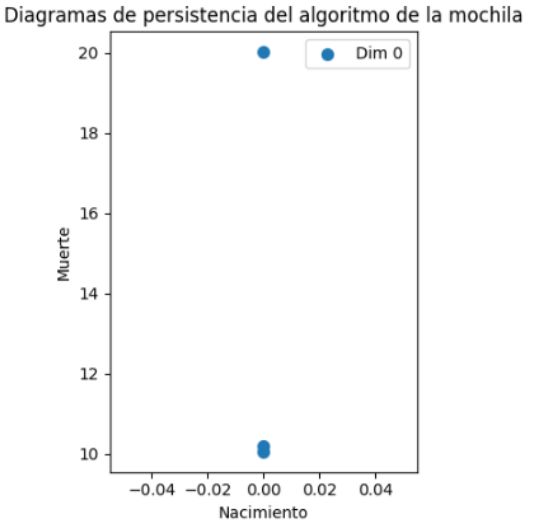

Figura 5.4.4*: Diagrama de Persistencia en 2D*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 2 dimensiones), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escalas.

5.5 Método Fuerza Bruta en Python

El método que se detalla a continuación es la “Fuerza Bruta”, el cual se encuentra descrito en Google Colab.

El método de la fuerza Bruta es una estrategia de solución que prueba todas las posibles combinaciones de elementos en la mochila para encontrar la combinación óptima. En el problema de la mochila, esto significa que se evalúan todas las posibles combinaciones de elementos para determinar cuál es la combinación que da la solución óptima.

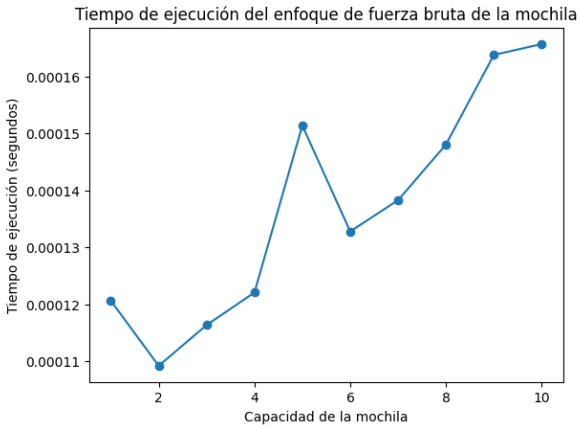

Figura 5.5: *Tiempo de ejecución respecto a la capacidad de la mochila.*

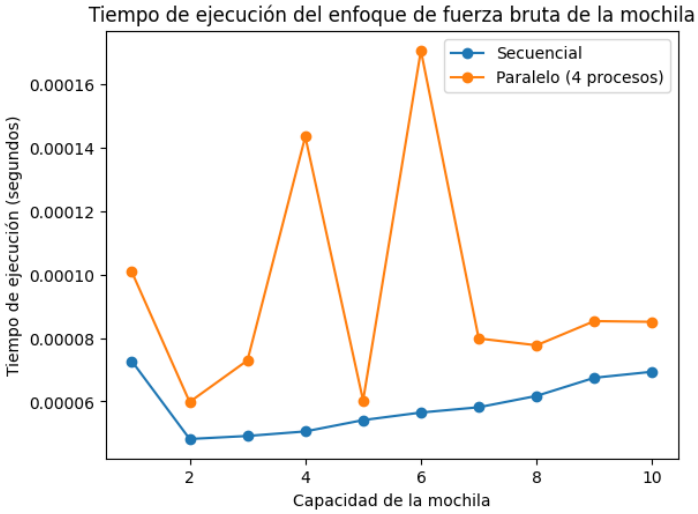

Figura 5.5.2: *Tiempo de ejecución respecto a la capacidad de la mochila.*

Este método puede llevar a la confusión con el método Greedy, pero se diferencian principalmente en que el método de fuerza bruta prueba todas las combinaciones posibles, mientras que el método Greedy toma decisiones localmente óptimas en cada paso. El método de fuerza bruta garantiza la solución óptima pero puede ser muy lento, mientras que el método Greedy es más rápido pero puede no proporcionar la solución óptima.

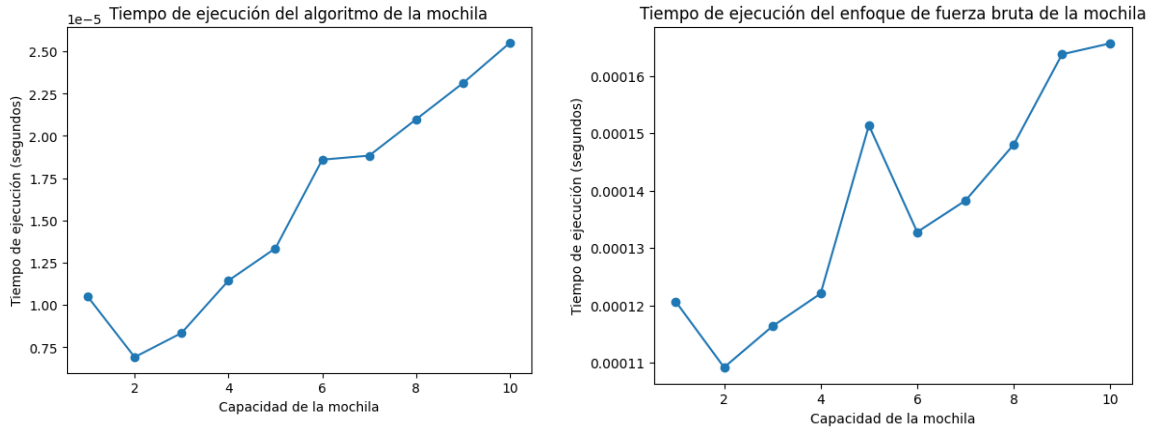

Figura 5.5.3: *Comparación gráfica entre métodos greedy y fuerza bruta.*

En esta situación, utilizamos la biblioteca de “multiprocessing” para crear código paralelo.

En este caso, el código secuencial es más rápido que el código paralelo en términos de velocidad de ejecución, ya que la estructura de datos que se está ocupando es de una cantidad pequeña y limitada.

Aplicando el modelado TDA, el análisis muestra la siguiente resolución:

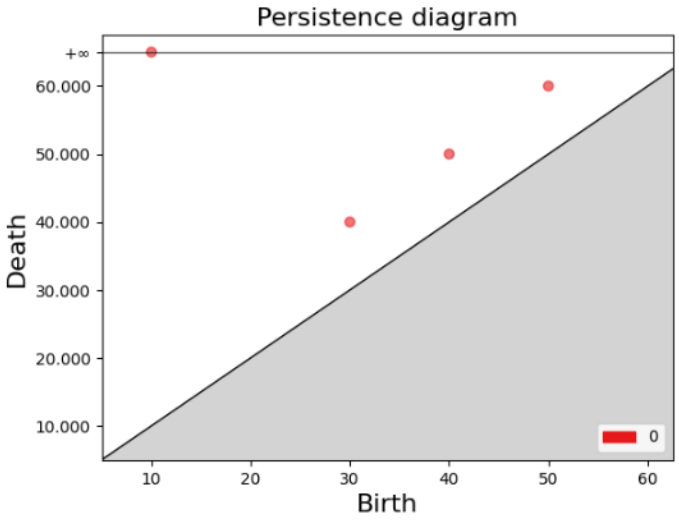

Figura 5.5.4: *Diagrama de persistencia algoritmo 2D*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 2 dimensiones), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escalas

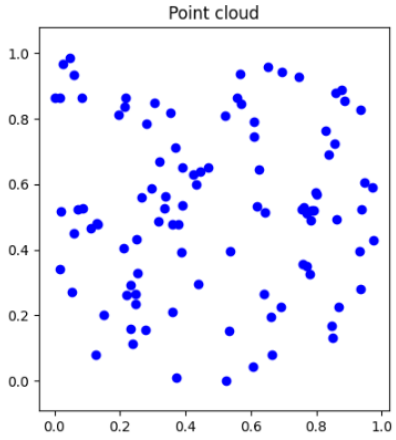

Figura 5.5.5: *Nube de Puntos*

En cuanto a las nubes de puntos, estas representan los datos en sí mismos. Cada punto en la nube representa un elemento de los datos y su posición en el espacio corresponde a sus características de peso o atributos.

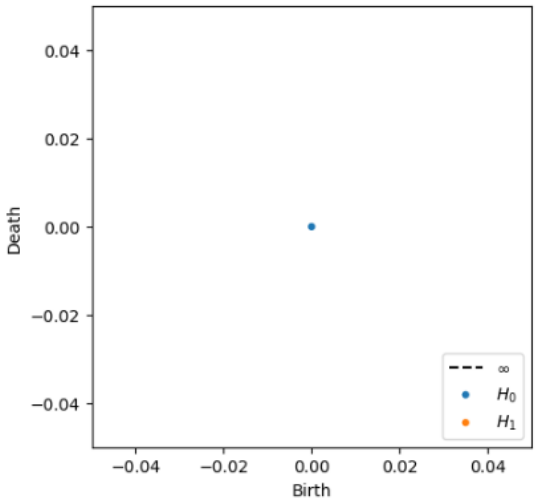

Figura 5.5.6: *Diagrama de Persistencia en 2D.*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 2 dimensiones), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escala.

5.6 Experimentación con pesos grandes

En esta ocasión, se realizarán pruebas con pesos al azar entre 1000 y 10000, variando cada peso entre 70 unidades para todos los casos previamente vistos.

Método Programación Lineal

Primero, podrán observar el gráfico de Programación lineal, el cuál muestra claramente el tiempo computacional que le toma al código realizar su proceso. En este caso, el algoritmo paralelo es más eficiente que el secuencial, al tener en su disposición mayor Núcleos de trabajo de CPU.

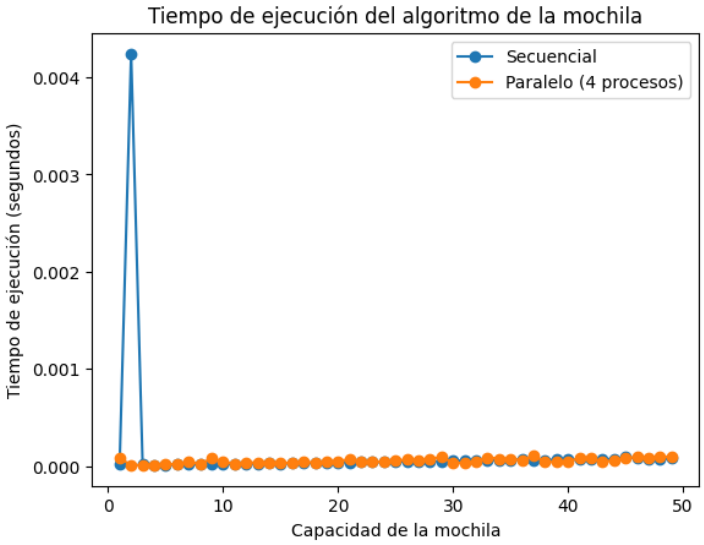

Figura 5.6: *Tiempo de ejecución respecto a la capacidad de la mochila Programación Dinámica.*

Esto demuestra, la eficiencia y la constancia de este algoritmo, ya que sus tiempos de ejecución son demasiado rápidos para este caso y permanecen en un margen claro, mientras que el algoritmo secuencial le cuesta iniciar, pero después le sigue la pulsada al algoritmo paralelo.

Aplicación de Modelización Secuencial en Julia:

```
Max 5 x[1] + 3 x[2] + 2 x[3] + 7 x[4] + 4 x[5]
Subject to
 2 x[1] + 8 x[2] + 4 x[3] + 2 x[4] + 5 x[5] <= 10.0
 x[1] binary
 x[2] binary
 x[3] binary
 x[4] binary
 x[5] binary
Running HiGHS 1.5.1 [date: 1970-01-01, git hash: 93f1876e4]
Copyright (c) 2023 HiGHS under MIT licence terms
```

```
Presolving model
1 rows, 5 cols, 5 nonzeros
1 rows, 4 cols, 4 nonzeros
Objective function is integral with scale 1

Solving MIP model with:
   1 rows
   4 cols (4 binary, 0 integer, 0 implied int., 0 continuous)
   4 nonzeros
```

```
Solving report
  Status            Optimal
  Primal bound      16
  Dual bound        16
  Gap               0% (tolerance: 0.01%)
  Solution status   feasible
                    16 (objective)
                    0 (bound viol.)
                    0 (int. viol.)
                    0 (row viol.)
  Timing            0.02 (total)
                    0.01 (presolve)
```

```
Nodes            1
LP iterations    1 (total)
                 0 (strong br.)
                 0 (separation)
                 0 (heuristics)
6.483837 seconds (8.02 M allocations: 421.814 MiB, 1.57% gc time, 99.04% compilation time: 1% of which was recompilation)
```

Aplicación de Modelización Secuencial en Julia

```
Objective is: 16.0
Solution is:
x[1] = 1, c[1]/w[1] = 2.5
x[2] = 0, c[2]/w[2] = 0.375
x[3] = 0, c[3]/w[3] = 0.5
x[4] = 1, c[4]/w[4] = 3.5
x[5] = 1, c[5]/w[5] = 0.8
  0.340985 seconds (1.22 M allocations: 63.473 MiB, 4.08% gc time, 98.35% compilation time)
```

Aplicación de Modelización Paralela en Julia:

```
0.009673 seconds (15.79 k allocations: 889.507 KiB, 56.31% compilation time)
    From worker 3:    Max 5 x[1] + 3 x[2] + 2 x[3] + 7 x[4] + 4 x[5]
    From worker 3:    Subject to
0.006371 seconds (6.96 k allocations: 373.486 KiB, 98.97% compilation time)
    From worker 2:    Max 5 x[1] + 3 x[2] + 2 x[3] + 7 x[4] + 4 x[5]
    From worker 2:    Subject to
    From worker 3:
```


Universidad Andrés Bello
Conectar · Innovar · Liderar

Se presentará a continuación el modelado topológico de datos (TDA)

Aplicando el modelado TDA, el análisis muestra la siguiente resolución:

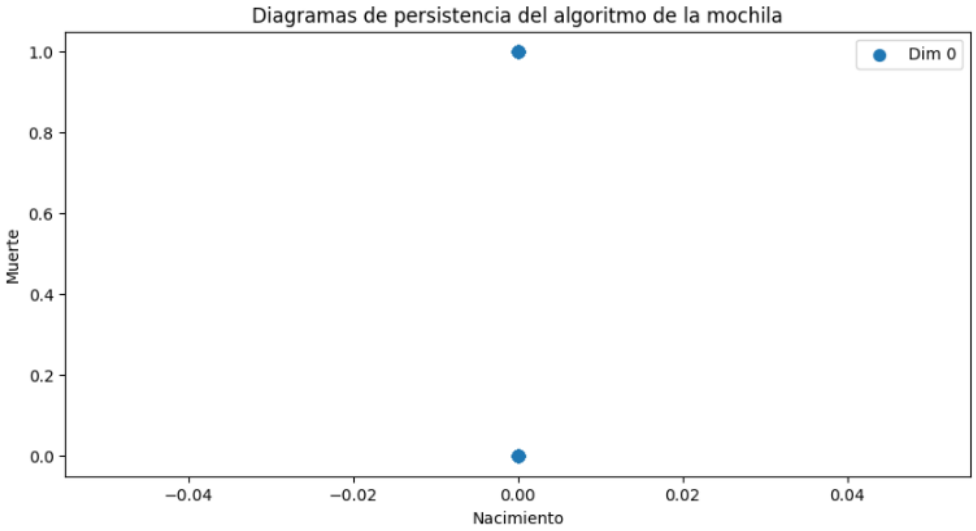

Figura 5.6.2: *Diagrama de persistencia algoritmo 2D*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 1 dimension), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escalas

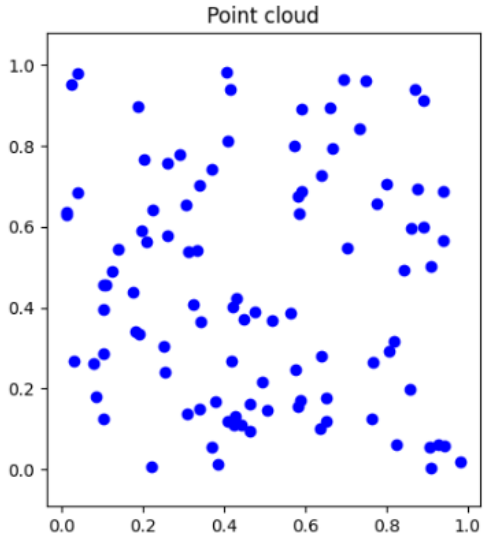

Figura 5.6.3: *Nube de Puntos*

En cuanto a las nubes de puntos, estas representan los datos en sí mismos. Cada punto en la nube representa un elemento de los datos y su posición en el espacio corresponde a sus características de peso o atributos.

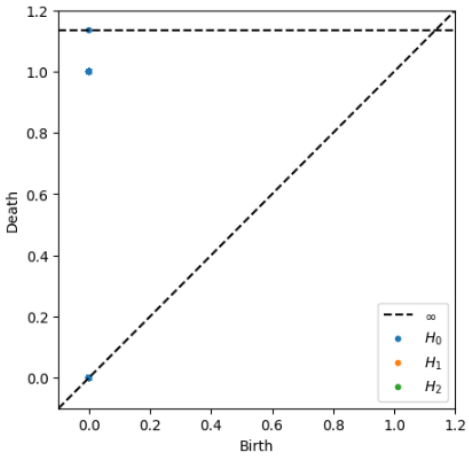

Figura 5.6.4: *Diagrama de Persistencia 2D*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 2 dimensiones), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escalas.

Método Greedy

Segundo, podrán observar el gráfico del método Greedy, el cuál muestra claramente el tiempo computacional que le toma al código realizar su proceso. En este caso, el algoritmo paralelo es más eficiente que el paralelo, ya que el algoritmo greedy, no siempre producir la solución óptima para el problema de la mochila, y puede ser menos efectivo.

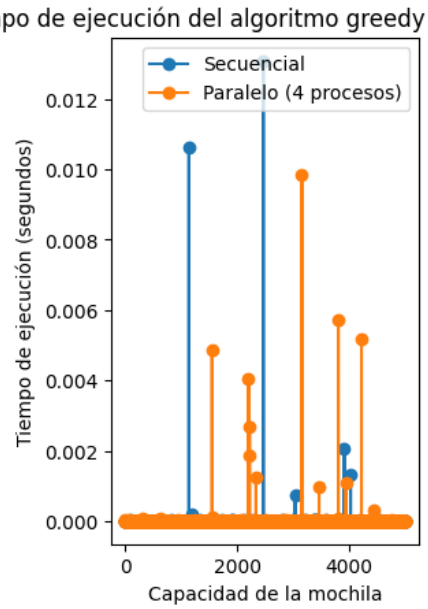

Figura 5.6.5: *Tiempo de ejecución respecto a la capacidad de la mochila método Greedy.*

Esto demuestra, la eficiencia de este algoritmo, ya que sus tiempos de ejecución son demasiado rápidos para este caso y permanecen en un margen claro.

Aplicación de Modelización Secuencial en Julia:

```
  0.123000 seconds (286.17 k allocations: 14.516 MiB, 21.06% gc time, 97.14% compilation time)
```

Aplicación de Modelización Paralela en Julia:

```
0.010271 seconds (15.79 k allocations: 889.507 KiB, 58.44% compilation time)
    From worker 3:    Max 5 x[1] + 3 x[2] + 2 x[3] + 7 x[4] + 4 x[5]
    From worker 3:    Subject to
    From worker 2:    Max 5 x[1] + 3 x[2] + 2 x[3] + 7 x[4] + 4 x[5]
    From worker 2:    Subject to
0.007588 seconds (6.96 k allocations: 373.486 KiB, 99.12% compilation time)
    From worker 3:
```

Se presentará a continuación el modelado topológico de datos (TDA)

Aplicando el modelado TDA, el análisis muestra la siguiente resolución:

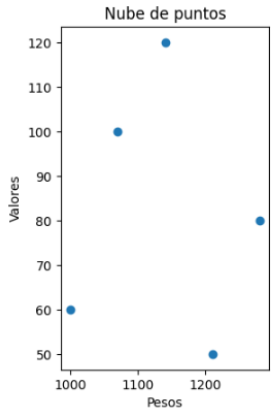

Figura 5.6.6: *Nube de Puntos*

En cuanto a las nubes de puntos, estas representan los datos en sí mismos. Cada punto en la nube representa un elemento de los datos y su posición en el espacio corresponde a sus características de peso o atributos.

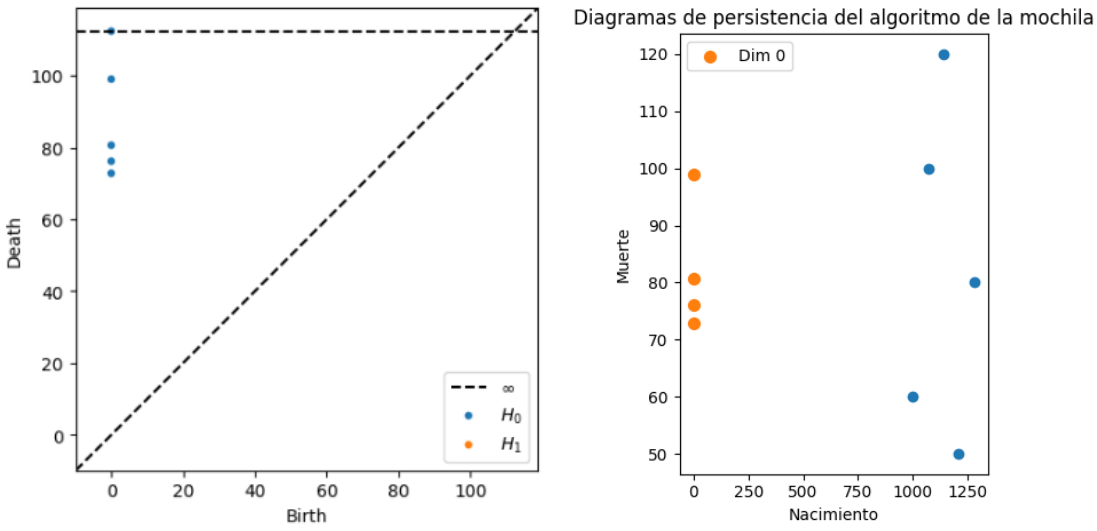

Figura 5.6.7: *Diagrama de Persistencia en 2D vista de dos maneras distintas*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 2 dimensiones), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escalas.

Método Fuerza Bruta

Tercero, podrán observar el gráfico del método de Fuerza Bruta, el cuál muestra claramente el tiempo computacional realizo en código al realizar su proceso.

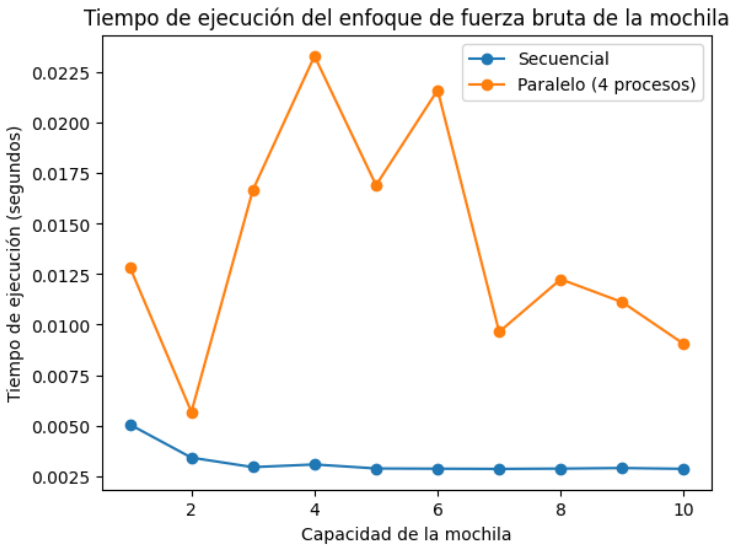

Figura 5.6.8*: Tiempo de ejecución respecto a la capacidad de la mochila Método Fuerza Bruta.*

Esto demuestra, la eficiencia de este algoritmo, ya que sus tiempos de ejecución son demasiado rápidos para este caso y permanecen en un margen claro.

Aplicación de Modelización Secuencial en Julia:

```
0.089874 seconds (8.02 M allocations: 421.814 MiB, 1.55% gc time, 99.14% compilation time: 1% of which was recompilation)
```

Aplicación de Modelización Paralela en Julia:

```
0.029342 seconds (15.79 k allocations: 889.507 KiB, 54.58% compilation time)
From worker 3:    Max 5 x[1] + 3 x[2] + 2 x[3] + 7 x[4] + 4 x[5]
From worker 3:    Subject to
From worker 2:    Max 5 x[1] + 3 x[2] + 2 x[3] + 7 x[4] + 4 x[5]
From worker 2:    Subject to
0.069542 seconds (6.96 k allocations: 373.486 KiB, 98.43% compilation time)
```

Aplicando el modelado TDA, el análisis muestra la siguiente resolución:

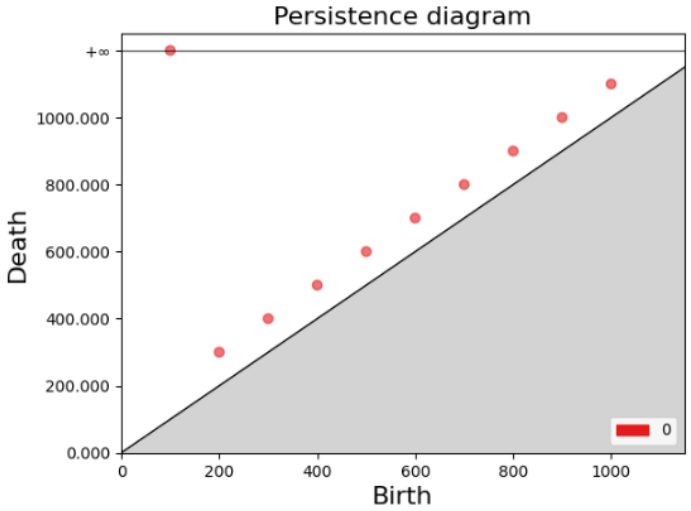

Figura 5.6.9: *Diagrama de persistencia algoritmo 2D*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 2 dimensiones), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escalas

En cuanto a las nubes de puntos, estas representan los datos en sí mismos. Cada punto en la nube representa un elemento de los datos y su posición en el espacio corresponde a sus características de peso o atributos.

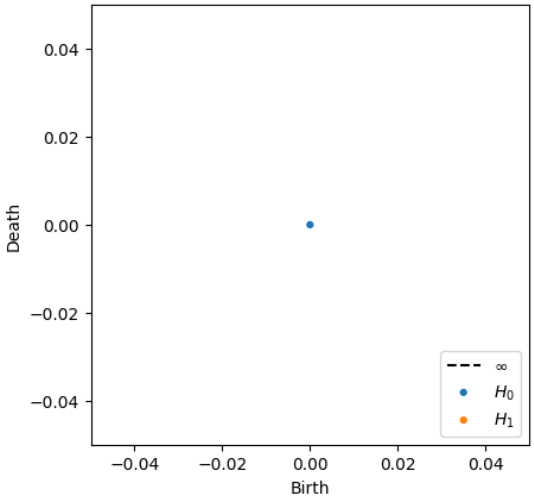

Figura 5.6.10: *Diagrama de Persistencia en 2D*

En el caso de este diagrama de persistencia, representan los componentes topológicos de los datos en diferentes dimensiones (en este caso son en 2 dimensiones), es decir, nos muestran cuáles son las regiones en el espacio donde los datos se agrupan y cuánto tiempo persisten esas regiones a lo largo de diferentes escala.

5.7 Heurística 0/1 implementada

La heurística presentada es conocida como el algoritmo de la mochila 0/1 (0/1 knapsack problem). Este problema se refiere a la optimización de la selección de elementos para llenar una mochila, teniendo en cuenta que cada elemento tiene un valor y un peso, y la mochila tiene una capacidad máxima.

El objetivo del algoritmo es determinar la combinación de elementos que maximice el valor total dentro de la mochila, sin exceder su capacidad. La restricción principal es que cada elemento solo puede ser seleccionado una vez (0/1), es decir, o se incluye completamente en la mochila o se excluye por completo y finalmente, se devuelve la solución obtenida y el valor máximo alcanzado.

En el código, se presentan diferentes instancias del problema como ejemplos. Cada instancia se compone de una lista de valores, una lista de pesos y una capacidad máxima de la mochila. El algoritmo se ejecuta para cada instancia y se mide el tiempo de ejecución.

Además, se realiza una gráfica que muestra la relación entre la capacidad de la mochila y el tiempo de ejecución, esto permite visualizar cómo varía el rendimiento del algoritmo en función de la capacidad.

Concluyendo, el algoritmo de la mochila 0/1 es una heurística eficiente para resolver el problema de optimización de selección de elementos en una mochila, maximizando el valor total sin exceder la capacidad. La implementación en el código utiliza programación dinámica y se presenta con ejemplos y una representación gráfica para analizar su desempeño.

Resultado por pantalla:

```
Instancia:
Valores: [60, 100, 120, 70, 40]
Pesos: [10, 20, 30, 40, 50]
Capacidad: 100
Tiempo de ejecución: 0.016669034957885742
Mejor solución: [0, 1, 2, 3]
Mejor valor: 350
```

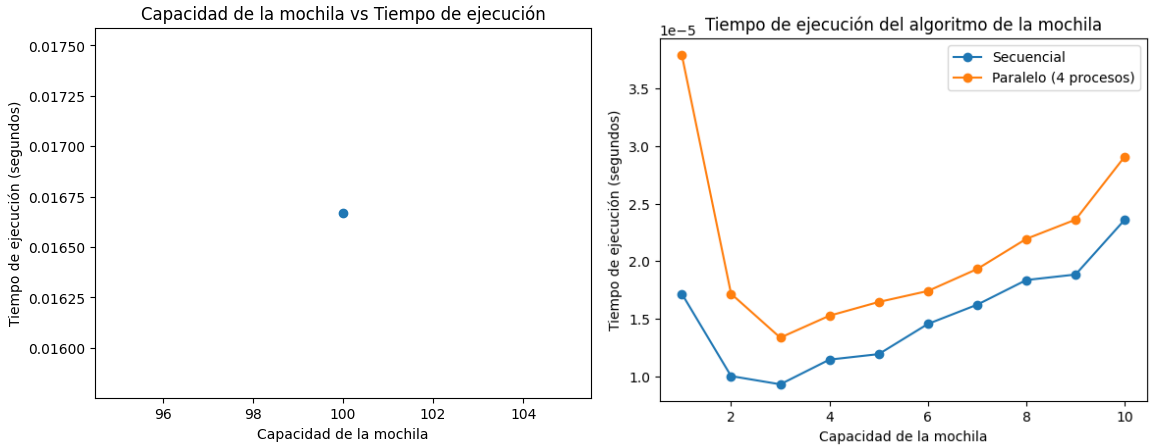

Figura 5.7: *Tiempo de ejecución en la solución óptima heurística 0/1 respecto al tiempo secuencial en menos capacidad.*

5.8 Resumen de la Experimentación

Para realizar estas comparaciones, se utilizaron diversas medidas de evaluación, como la calidad de la solución, el tiempo de ejecución y el número de iteraciones La calidad de la solución se mide en términos de la diferencia entre el valor obtenido por el algoritmo y el valor óptimo conocido para el conjunto de datos. El tiempo de ejecución se mide en segundos. El número de iteraciones es una medida del número de veces que el algoritmo accede al conjunto de datos.

Los experimentos computacionales también pueden incluir análisis estadísticos para determinar si las diferencias en el rendimiento de los algoritmos son estadísticamente significativas. Esto se hizo mediante pruebas de hipótesis y análisis de varianza.

Al hacer el análisis de varianza y el cálculo de la prueba de hipótesis se deduce que para los 3 casos los valores de F y p serán los mismos, y que al realizar el ANOVA, se dicta que “No hay suficiente evidencia para rechazar la hipótesis nula de que no hay diferencia significativa entre las versiones secuencial y paralela”.

Tras tener toda esta información, podemos determinar que:

- La fuerza bruta es el método más ineficiente en términos de tiempo, ya que examina todas las combinaciones posibles de elementos. Es adecuado para problemas pequeños, pero no para problemas más grandes.
- La programación dinámica divide el problema en subproblemas más pequeños, lo que puede ser muy eficiente. Sin embargo, el tiempo de ejecución también puede aumentar exponencialmente con el número de elementos.
- El método greedy elige los elementos con la mejor relación valor-peso en cada iteración. Es más rápido que la programación dinámica, pero no siempre garantiza la solución óptima, especialmente cuando los elementos tienen valores muy dispares.

El análisis combinado del diagrama de persistencia y la nube de puntos se erige como una técnica fundamental para el descubrimiento de patrones y estructuras subyacentes en datos complejos.

El diagrama de persistencia permite la visualización de la evolución temporal de los componentes topológicos, mientras que la nube de puntos muestra la distribución de los datos en el espacio.

La combinación de ambas técnicas proporciona una comprensión profunda de la estructura de los datos y de las relaciones entre los elementos que los conforman. Por tanto, el modelado topológico de datos (TDA) se presenta como una herramienta poderosa y valiosa para el análisis y la comprensión de datos complejos en diversas áreas de investigación.

En este sentido, es importante tener en cuenta las reglas de ética académica y de citación de fuentes confiables para garantizar la originalidad y la calidad del trabajo de investigación.

En resumen, los experimentos computacionales han demostrado que algunos algoritmos funcionan mejor que otros en diferentes tipos de instancias del problema de la mochila. Por lo tanto, el método más adecuado dependerá del tamaño y los límites del problema específico, y cada algoritmo tiene sus ventajas y desventajas.

5.9 Tabla de recursos

Se presentará a continuación la Tabla de recursos de cada algoritmo.

Donde:

- **Instancia:** Etapa del proceso del algoritmo
- **Capacidad:** Capacidad Usada del total calculada tras los resultados del algoritmo.
- **T.Secuencial:** Tiempo utilizado con un solo núcleo de CPU para realizar la ejecución según la instancia medida en segundos.
- **T.Paralelo:** Tiempo utilizado con múltiples núcleos de CPU (4 núcleos para este caso) para realizar la ejecución según la instancia medida en segundos

Método Programación Lineal

Ins	Capacidad\T. secuencial		T. paralelo
1	1	1.3828277587890625e-05	3.838539123535156e-05
2	1	8.106231689453125e-06	9.5367431640625e-06
3	1	9.059906005859375e-06	9.298324584960938e-06
4	1	1.0251998901367188e-05	1.2874603271484375e-05
5	1	1.1444091796875e-05	4.649162292480469e-05
6	1	0.0003006458282470703	1.9073486328125e-05
7	1	2.002716064453125e-05	1.9550323486328125e-05
8	1	2.7894973754882812e-05	2.2649765014648438e-05
9	1	1.7642974853515625e-05	2.5033950805664062e-05
10	1	1.9550323486328125e-05	4.434585571289625e-05
11	10	2.09808349609375e-05	2.6941299438476562e-05
12	10	2.1696090698242188e-05	2.8371810913085938e-05
13	10	0.0002319812774658203	3.0994415283203125e-05
14	10	3.0040740966796875e-05	3.266334533691406e-05
15	10	2.765655517578125e-05	3.3855438232421875e-05
16	10	2.765655517578125e-05	4.2438507080078125e-05
17	10	2.765655517578125e-05	3.8623809814453125e-05
18	10	3.2901763916015625e-05	4.673004150390625e-05
19	10	3.743171691894531e-05	4.124641418457031e-05
20	10	3.719329833984375e-05	4.1961669921875e-05
21	20	3.910064697265625e-05	4.5299530029296875e-05
22	20	4.3392181396484375e-05	6.031990051269531e-05
23	20	5.1975250244140625e-05	4.839897155761719e-05
24	20	4.8160552978515625e-05	5.054473876953125e-05
25	20	4.839897155761719e-05	5.888938903808594e-05
26	20	5.054473876953125e-05	6.198883056640625e-05
27	20	4.863739013671875e-05	5.5789947509765625e-05
28	20	6.198883056640625e-05	7.43865966796875e-05
29	20	5.125999450683594e-05	6.008148193359375e-05
30	20	4.792213439941406e-05	6.151199340820312e-05
31	30	0.0002930164337158203	6.151199340820312e-05
32	30	9.489059448242188e-05	6.961822509765625e-05
33	30	5.602836608886719e-05	6.771087646484375e-05
34	30	0.00027680397033691406	6.914138793945312e-05
35	30	6.031990051269531e-05	7.104873657226562e-05
36	30	6.604194641113281e-05	7.939338684082031e-05
37	30	9.131431579589844e-05	7.486343383789062e-05
38	30	7.033348083496094e-05	0.00019240379333496094
39	30	6.914138793945312e-05	0.0019102096557617188
40	30	7.653236389160156e-05	8.535385131835938e-05
41	40	7.653236389160156e-05	0.00011157989501953125
42	40	7.748603820800781e-05	8.916854858398438e-05
43	40	6.890296936035156e-05	8.487701416015625e-05
44	40	6.794929504394531e-05	9.34600830078125e-05
45	40	6.413459777832031e-05	9.72747802734375e-05
46	40	0.0002677440643310547	0.00013828277587890625
47	40	0.00010609626770019531	9.083747863769531e-05
48	40	7.319450378417969e-05	0.0001010894775390625
49	49	7.319450378417969e-05	9.417533874511719e-05

Tabla 5.9: *Instancias, capacidad y tiempo Método Programación lineal y Heurística*

Universidad Andrés Bello
Conectar · Innovar · Liderar

Método Greedy

Ins	Capacidad\T. secuencial		T. paralelo
1	10	1.621246337890625e-05	7.319450378417969e-05
2	10	4.76837158203125e-06	7.3909759521484375e-06
3	10	4.291534423828125e-06	5.245208740234375e-06
4	10	3.814697265625e-06	4.76837158203125e-06
5	10	3.5762786865234375e-06	3.814697265625e-06
6	10	3.814697265625e-06	3.814697265625e-06
7	10	3.5762786865234375e-06	3.814697265625e-06
8	10	3.814697265625e-06	4.291534423828125e-06
9	10	3.5762786865234375e-06	4.291534423828125e-06
10	10	3.814697265625e-06	4.5299530029296875e-06
11	10	3.814697265625e-06	3.814697265625e-06
12	10	3.814697265625e-06	3.814697265625e-06
13	10	3.5762786865234375e-06	4.291534423828125e-06
14	10	3.5762786865234375e-06	4.5299530029296875e-06
15	10	3.5762786865234375e-06	4.291534423828125e-06
16	10	3.5762786865234375e-06	4.0531158447265625e-06
17	10	3.814697265625e-06	4.291534423828125e-06
18	10	3.5762786865234375e-06	4.5299530029296875e-06
19	10	3.5762786865234375e-06	4.291534423828125e-06
20	10	3.5762786865234375e-06	4.291534423828125e-06
21	10	3.814697265625e-06	3.814697265625e-06
22	10	3.5762786865234375e-06	1.3113021850585938e-05
23	10	3.814697265625e-06	4.0531158447265625e-06
24	10	3.5762786865234375e-06	3.814697265625e-06
25	10	3.5762786865234375e-06	3.814697265625e-06
26	10	3.814697265625e-06	4.291534423828125e-06
27	10	3.814697265625e-06	4.291534423828125e-06
28	10	3.5762786865234375e-06	4.0531158447265625e-06
29	10	4.5299530029296875e-06	3.814697265625e-06
30	10	3.814697265625e-06	4.291534423828125e-06
31	10	3.814697265625e-06	4.291534423828125e-06
32	10	3.5762786865234375e-06	4.291534423828125e-06
33	10	3.5762786865234375e-06	4.76837158203125e-06
34	10	3.5762786865234375e-06	4.0531158447265625e-06
35	10	3.814697265625e-06	4.76837158203125e-06
36	10	3.5762786865234375e-06	4.291534423828125e-06
37	10	3.5762786865234375e-06	4.291534423828125e-06
38	10	4.0531158447265625e-06	4.5299530029296875e-06
39	10	3.5762786865234375e-06	4.0531158447265625e-06
40	10	3.337860107421875e-06	4.291534423828125e-06
41	10	3.5762786865234375e-06	4.5299530029296875e-06
42	10	3.5762786865234375e-06	4.291534423828125e-06
43	10	3.5762786865234375e-06	4.291534423828125e-06
44	10	3.5762786865234375e-06	4.291534423828125e-06
45	10	3.5762786865234375e-06	4.291534423828125e-06
46	10	3.814697265625e-06	4.5299530029296875e-06
47	10	3.5762786865234375e-06	3.814697265625e-06
48	10	3.337860107421875e-06	4.291534423828125e-06
49	10	3.5762786865234375e-06	4.291534423828125e-06

Tabla 5.9.1: *Instancias, capacidad y tiempo Método Greedy*

Universidad Andrés Bello
Conectar · Innovar · Liderar

Ins	Capacidad\T. secuencial		T. paralelo
4949	5000	4.76837158203125e-06	3.0994415283203125e-06
4950	5000	5.4836273193359375e-06	3.0994415283203125e-06
4951	5000	5.0067901611328125e-06	3.0994415283203125e-06
4952	5000	4.76837158203125e-06	3.0994415283203125e-06
4953	5000	5.7220458984375e-06	3.0994415283203125e-06
4954	5000	5.7220458984375e-06	3.0994415283203125e-06
4955	5000	5.4836273193359375e-06	3.337860107421875e-06
4956	5000	5.0067901611328125e-06	3.337860107421875e-06
4957	5000	5.4836273193359375e-06	3.0994415283203125e-06
4958	5000	6.198883056640625e-06	2.86102294921875e-06
4959	5000	4.76837158203125e-06	2.86102294921875e-06
4960	5000	5.4836273193359375e-06	2.86102294921875e-06
4961	5000	5.7220458984375e-06	3.0994415283203125e-06
4962	5000	5.0067901611328125e-06	3.337860107421875e-06
4963	5000	5.7220458984375e-06	3.0994415283203125e-06
4964	5000	5.7220458984375e-06	3.337860107421875e-06
4965	5000	5.7220458984375e-06	3.0994415283203125e-06
4966	5000	5.245208740234375e-06	3.0994415283203125e-06
4967	5000	5.4836273193359375e-06	3.337860107421875e-06
4968	5000	5.7220458984375e-06	3.0994415283203125e-06
4969	5000	5.4836273193359375e-06	3.0994415283203125e-06
4970	5000	5.245208740234375e-06	3.337860107421875e-06
4971	5000	5.245208740234375e-06	3.0994415283203125e-06
4972	5000	5.245208740234375e-06	3.0994415283203125e-06
4973	5000	5.0067901611328125e-06	3.0994415283203125e-06
4974	5000	5.245208740234375e-06	3.0994415283203125e-06
4975	5000	5.7220458984375e-06	3.0994415283203125e-06
4976	5000	5.245208740234375e-06	3.0994415283203125e-06
4977	5000	5.245208740234375e-06	3.0994415283203125e-06
4978	5000	5.4836273193359375e-06	3.337860107421875e-06
4979	5000	5.245208740234375e-06	3.0994415283203125e-06
4980	5000	5.245208740234375e-06	3.0994415283203125e-06
4981	5000	5.7220458984375e-06	3.0994415283203125e-06
4982	5000	5.4836273193359375e-06	3.0994415283203125e-06
4983	5000	4.76837158203125e-06	3.337860107421875e-06
4984	5000	5.245208740234375e-06	3.0994415283203125e-06
4985	5000	5.245208740234375e-06	3.0994415283203125e-06
4986	5000	5.7220458984375e-06	3.0994415283203125e-06
4987	5000	5.7220458984375e-06	3.0994415283203125e-06
4988	5000	5.960464477539062e-06	3.0994415283203125e-06
4989	5000	5.245208740234375e-06	3.337860107421875e-06
4990	5000	5.245208740234375e-06	3.0994415283203125e-06
4991	5000	5.4836273193359375e-06	3.0994415283203125e-06
4992	5000	5.245208740234375e-06	3.0994415283203125e-06
4993	5000	5.4836273193359375e-06	3.0994415283203125e-06
4994	5000	5.4836273193359375e-06	3.0994415283203125e-06
4995	5000	5.4836273193359375e-06	3.5762786865234375e-06
4996	5000	5.0067901611328125e-06	3.0994415283203125e-06
4997	5000	5.4836273193359375e-06	3.0994415283203125e-06
4998	5000	5.245208740234375e-06	2.86102294921875e-06
4999	5000	5.0067901611328125e-06	2.86102294921875e-06

Tabla 5.9.1: *Continuación Instancias, capacidad y tiempo Método Greedy.*

Universidad Andrés Bello
Conectar · Innovar · Liderar

Método Fuerza Bruta

Ins	Capacidad\T. secuencial		T. paralelo
1	1000	0.004990577697753906	0.005790233612060547
2	1000	0.00542140007019043	0.005839824676513672
3	1000	0.004929780960083008	0.0058746337890625
4	1000	0.0034546852111816406	0.0057582855224609375
5	1000	0.002913236618041992	0.011101245880126953
6	1000	0.0029494762420654297	0.0058896541595458984
7	1000	0.0029480457305908203	0.013058185577392578
8	1000	0.002920866012573242	0.012908935546875
9	1000	0.002949237823486328	0.012909173965454102
10	1000	0.002964496612548828	0.017188549041748047
11	1000	0.0028998851776123047	0.005927324295043945
12	1000	0.002910614013671875	0.012996673583984375
13	1000	0.002896547317504883	0.012981653213500977
14	1000	0.003160238265991211	0.0068187713623046875
15	1000	0.0029664039611816406	0.01296854019165039
16	1000	0.002924680709838867	0.011954545974731445
17	1000	0.0030164718627929688	0.022681236267089844
18	1000	0.0028934478759765625	0.018175601959228516
19	1000	0.002886056900024414	0.011124610900878906
20	1000	0.0028829574584960938	0.012945890426635742
21	1000	0.0029306411743164062	0.012970685958862305
22	1000	0.0030167102813720703	0.01301431655883789
23	1000	0.0031328201293945312	0.01601266860961914
24	1000	0.005046367645263672	0.005887031555175781
25	1000	0.005265474319458008	0.012908935546875
26	1000	0.0054187774658203125	0.021384716033935547
27	1000	0.005627870559692383	0.005316019058227539
28	1000	0.003770112991333008	0.02346181869506836
29	1000	0.0029473304748535156	0.01591968536376953
30	1000	0.0029172897338867188	0.015952110290527344
31	1000	0.0029327869415283203	0.012912511825561523
32	1000	0.002863645553588867	0.012428045272827148
33	1000	0.002867460250854492	0.0179293155670166
34	1000	0.0028815269470214844	0.012935876846313477
35	1000	0.0029015541076660156	0.012912273406982422
36	1000	0.002854585647583008	0.010943174362182617
37	1000	0.0029845237731933594	0.006105899810791016
38	1000	0.002889394760131836	0.012941122055053711
39	1000	0.002863168716430664	0.005974531173706055
40	1000	0.0030019283294677734	0.012513875961303711
41	1000	0.0029058456420898438	0.014430522918701172
42	1000	0.002872467041015625	0.0058972835540771484
43	1000	0.002851247787475586	0.017888784408569336
44	1000	0.0028765201568603516	0.012868404388427734
45	1000	0.0028781890869140625	0.012878894805908203
46	1000	0.0028722286224365234	0.012888669967651367
47	1000	0.0028884410858154297	0.008631229400634766
48	1000	0.0029108524322509766	0.012921810150146484
49	1000	0.0028760433197021484	0.005948781967163086

Tabla 5.9.2: *Instancias, capacidad y tiempo Método Fuerza Bruta.*

Universidad Andrés Bello
Conectar · Innovar · Liderar

Ins	Capacidad\T. secuencial		T. paralelo
4948	5000	0.0056192874908447266	0.011004924774169922
4949	5000	0.005584239959716797	0.011432647705078125
4950	5000	0.005599260330200195	0.011422395706176758
4951	5000	0.005716562271118164	0.011884450912475586
4952	5000	0.005640268325805664	0.011358499526977539
4953	5000	0.00560760498046875	0.011348247528076172
4954	5000	0.005635261535644531	0.011394739151000977
4955	5000	0.005574703216552734	0.01147770881652832
4956	5000	0.005567312240600586	0.011424779891967773
4957	5000	0.005696773529052734	0.011324882507324219
4958	5000	0.0056307315826416016	0.01138448715209961
4959	5000	0.005731821060180664	0.01136922836303711
4960	5000	0.005595684051513672	0.011415481567382812
4961	5000	0.006105661392211914	0.011326789855957031
4962	5000	0.006896495819091797	0.011391639709472656
4963	5000	0.005684852600097656	0.011317729949951172
4964	5000	0.0056073665618896484	0.011303186416625977
4965	5000	0.005612611770629883	0.01140737533569336
4966	5000	0.0055963993072509766	0.011456012725830078
4967	5000	0.005593776702880859	0.013193607330322266
4968	5000	0.007111787796020508	0.01144266128540039
4969	5000	0.008584260940551758	0.011349201202392578
4970	5000	0.005609750747680664	0.011321306228637695
4971	5000	0.005682706832885742	0.011367321014404297
4972	5000	0.005633831024169922	0.011422157287597656
4973	5000	0.005691051483154297	0.011263608932495117
4974	5000	0.005644083023071289	0.011462926864624023
4975	5000	0.005582571029663086	0.011346817016601562
4976	5000	0.0056231021881103516	0.011252880096435547
4977	5000	0.005556821823120117	0.011493444442749023
4978	5000	0.005540370941162109	0.01150369644165039
4979	5000	0.005544424057006836	0.011929035186767578
4980	5000	0.0057163238525390625	0.011521339416503906
4981	5000	0.005610466003417969	0.011407136917114258
4982	5000	0.005677700042724609	0.011372566223144531
4983	5000	0.005835771560668945	0.011321783065795898
4984	5000	0.0057065486907958984	0.00832509994506836
4985	5000	0.005593776702880859	0.0056116580963134766
4986	5000	0.005726337432861328	0.005568265914916992
4987	5000	0.005918264389038086	0.005613565444946289
4988	5000	0.00593876838684082	0.005541801452636719
4989	5000	0.005724191665649414	0.005555629730224609
4990	5000	0.005605220794677734	0.0055866241455078125
4991	5000	0.0056040287017822266	0.005571842193603516
4992	5000	0.0056285858154296875	0.005605220794677734
4993	5000	0.005606889724731445	0.005547046661376953
4994	5000	0.005655050277709961	0.005578517913818359
4995	5000	0.005766153335571289	0.005545377731323242
4996	5000	0.005613088607788086	0.006020307540893555
4997	5000	0.00561833381652832	0.0057027339935302734
4998	5000	0.006313323974609375	0.005909919738769531
4999	5000	0.007035255432128906	0.006028652191162109

Tabla 5.9.2: *Continuación Instancias, capacidad y tiempo Método Fuerza Bruta.*

Universidad Andrés Bello
Conectar · Innovar · Liderar

CAPÍTULO 6

PROBLEMA PLANTEADO EN CASO NACIONAL

En este punto, veremos este problema implementado en terreno nacional a través del servicio de Correos “Correos de chile” implementado desde el punto de vista de la optimización de rutas de entregas para su resolución con el problema de la mochila.

El problema de la mochila es una técnica que se puede utilizar en la optimización de rutas de entregas. Al implementar esta técnica, se determina qué paquetes deben cargarse en cada camión para maximizar la cantidad de entregas en un solo viaje y reducir los costos de transporte.

Cada paquete se considera como un objeto en el problema de la mochila, con un valor asociado que representa el ingreso esperado de su entrega, y un peso que representa el espacio que ocupa en el camión.

La tarea es seleccionar un conjunto de paquetes que puedan transportarse juntos en un solo viaje sin exceder la capacidad del camión. Al resolver el problema de la mochila y determinar el conjunto óptimo de paquetes para cargar en el camión, se pueden planificar las rutas de entrega para optimizar el tiempo y la distancia de viaje.

En resumen, el problema de la mochila es una herramienta muy útil para ayudar a las empresas a maximizar la eficiencia de sus entregas y reducir los costos de transporte.

CAPÍTULO 7

CONCLUSIONES

Al concluir este caso, podemos deducir que el problema de la mochila tiene una amplia gama de aplicaciones en la vida real, que van desde la selección de elementos para empacar en una maleta para un viaje, hasta la planificación de recursos en una empresa. La resolución de este problema puede proporcionar una solución óptima para elegir entre una gran cantidad de elementos, considerando tanto su valor como su peso, y optimizando el espacio y recursos disponibles.

El modelo matemático, los métodos algorítmicos y el solver utilizados en la experimentación computacional son herramientas poderosas que pueden adaptarse a diferentes tipos y tamaños de problemas. A través de la experimentación, se pudo evaluar la precisión y la eficiencia de la solución para diferentes conjuntos de datos, lo que permite obtener una mejor comprensión de la complejidad del problema y la capacidad del modelo para manejar situaciones reales.

En conclusión, la resolución del problema de la mochila a través de la experimentación computacional nos proporciona una herramienta efectiva para la toma de decisiones en situaciones de optimización combinatoria. La capacidad de aplicar este modelo en diferentes situaciones de la vida real lo hace una herramienta valiosa para la optimización de recursos y el ahorro de costos en una amplia variedad de contextos.

REFERENCIAS

[1] Pisinger, D. (2005). Where are the hard knapsack problems? Computers & Operations Research, 32(9), 2271-2284. https://www.dcs.gla.ac.uk/~pat/cpM/jchoco/knapsack/papers/hardInstances.pdf

[2] ChatGPT, OpenAI. (2023). Knapsack Solution with Felipe Pereira. https://chat.openai.com/

[3] AMPL Optimization LLC. (2021). AMPL: A Modeling Language for Mathematical Programming. https://ampl.com/

[4] Martello, S., & Toth, P. (1990). Knapsack problems: algorithms and computer implementations. John Wiley & Sons. https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Knapsack%20Problems_%20Algorithms%20and%20Computer%20Implementations%20%5BMartello%20%26%20Toth%201990-11%5D.pdf

[5] Universidad Autónoma del Estado de Hidalgo. (s.f.). Problema de la mochila. Tlahuelilpan, boletín informativo de la Dirección de Extensión Universitaria. https://www.uaeh.edu.mx/scige/boletin/tlahuelilpan/n6/e2.html

[6] Pereira, F. (2023). Knapsack problem by Felipe Pereira [Knapsack problem by Felipe Pereira.ipynb]. Recuperado de https://colab.research.google.com/drive/1JAYtKnKFLWvVITR42KaTVWCjBCcuM0Dt?usp=sharing

[7] Verdeguer D. & Alonso P. (2016). Solución en paralelo del problema de la mochila. Upv.es. Recuperado de https://riunet.upv.es/bitstream/handle/10251/87210/VERDEGUER%20-%20Soluci%C3%B3n%20en%20paralelo%20del%20problema%20de%20la%20mochila..pdf?sequence=1