Informe Técnico

Análisis de Complejidad y Diseño de Algoritmos para el Problema del Transporte Discreto

Discrete Logistics

Autores:
Adrian Alejandro Souto Morales
Gabriel Herrera Carrazana

2025

Proyecto Final de Diseño y Análisis de Algoritmos

1. Introducción

En *Discrete Logistics*, operamos en un nicho de mercado extremadamente delicado: el transporte transfronterizo de productos de valor incalculable y alta sensibilidad. Nuestros clientes confían en nosotros para mover artículos únicos y a menudo irremplazables, donde la discreción, la seguridad y la mitigación de riesgos son absolutamente críticas. La pérdida de una parte significativa de un envío puede tener consecuencias devastadoras.

Nos enfrentamos a un desafío logístico y estratégico fundamental en la preparación de cada operación de transporte internacional. Disponemos de un conjunto de artículos de alto valor (ej. obras de arte, documentos clasificados, prototipos tecnológicos) que necesitan ser movidos. Cada uno de estos artículos tiene un peso y un valor intrínseco que representa su importancia y el riesgo asociado a su pérdida.

Para realizar estos envíos, utilizamos una red de transportistas personales, a quienes llamamos "mulas". Cada mula tiene una capacidad máxima de peso que puede transportar discretamente como equipaje personal sin levantar sospechas.

El problema que necesitamos resolver es: ¿Cómo podemos seleccionar y distribuir los artículos disponibles entre las mulas designadas para el transporte, de modo que minimicemos el riesgo general del envío?

Esto implica tomar decisiones críticas bajo las siguientes condiciones:

1. Ninguna mula puede exceder su capacidad máxima de peso con los artículos que se le asignen.

2. Para maximizar la probabilidad de que, si una mula es interceptada o comprometida, la pérdida no sea catastrófica, necesitamos distribuir el valor total de los artículos transportados de la manera más equitativa posible entre las mulas. Es decir, queremos que la diferencia máxima en el valor total de los artículos contenidos en el equipaje de cualquier par de mulas sea la menor posible.

3. Hay una cantidad máxima de mulas disponibles.

Una distribución desequilibrada del valor entre las mulas aumentaría drásticamente el riesgo de una pérdida masiva si la mula con la carga "más valiosa" fuera comprometida. Necesitamos una estrategia de empaquetado que balancee el valor de forma óptima, respetando las limitaciones de peso de cada transportista y utilizando el número exacto de mulas requerido.Si no es posible cumplir con estas condiciones para el conjunto de artículos y mulas dado, necesitamos saberlo.

1.1. Justificación del Análisis de Complejidad

Realizar un análisis de complejidad es un paso estratégico fundamental antes de acometer el diseño de algoritmos. Este análisis nos permite determinar la intratabilidad inherente de un problema y, por tanto, establecer si es factible encontrar soluciones óptimas en un tiempo de ejecución razonable. Para el problema del Transporte Discreto, este análisis determina si Discrete Logistics puede garantizar a sus clientes una distribución de riesgo probadamente óptima, o si debe confiar en soluciones "suficientemente buenas" que sean computacionalmente rápidas. La conclusión de este análisis justificará la necesidad de explorar algoritmos de aproximación o heurísticas eficientes.

2. Definición Formal del Problema

Dado un conjunto de N artículos $A = \{a_1, ..., a_n\}$ y M mulas $M = \{m_1, ..., m_m\}$, buscamos una partición tal que:

1. **Restricción de Capacidad:** $\sum_{a_i \in A_j} p_i \leq C_j$ para toda mula m_j.
2. **Función Objetivo (Equidad):** Minimizar K tal que $|V_j - V_k| \leq K$ para todo par de mulas (j, k), donde V_j es el valor total asignado a la mula j.

3. Análisis de Complejidad (Fase 2)

Para determinar la viabilidad computacional, analizamos la complejidad teórica del problema.

3.1. Pertenencia a la Clase NP

Argumentamos que el problema de decisión del Transporte Discreto pertenece a la clase de complejidad **NP** (Nondeterministic Polynomial time). Un problema está en NP si una solución candidata puede ser verificada en tiempo polinomial.

Para demostrarlo, definimos un algoritmo verificador que opera sobre un "certificado". El certificado, en este caso, puede ser representado como un array de N enteros, $Asignacion[1..N]$, donde $Asignacion[i] = j$ significa que el artículo a_i se asigna a la mula m_j.

Algoritmo Verificador:
Dada una asignación candidata (el certificado):

1. **Verificación de Capacidad (Tiempo Polinomial):**
   - Para cada una de las M mulas, inicializar un *peso_actual* en 0.
   - Iterar sobre los N artículos. Para cada artículo a_i, sumar su peso p_i al *peso_actual* de la mula a la que ha sido asignado.
   - Después de procesar todos los artículos, comprobar para cada una de las M mulas si su *peso_actual* es menor o igual a su capacidad C_j.
   - Este proceso implica N sumas y M comparaciones, lo cual tiene una complejidad de $O(N + M)$.
2. **Verificación de Equidad de Valor (Tiempo Polinomial):**
   - Para cada una de las M mulas, calcular el valor total de los artículos asignados, $V_j = \sum_{a_i \in A_j} v_i$. Esto requiere $O(N)$ operaciones.
   - Iterar sobre todos los pares de mulas (m_j, m_k). Hay $M(M-1)/2$ pares.
   - Para cada par, calcular la diferencia de valor $|V_j - V_k|$ y comprobar si es mayor que K. Si alguna diferencia excede K, la condición falla.
   - Este proceso tiene una complejidad de $O(M^2)$.

Dado que ambas verificaciones se pueden realizar en un tiempo polinomial ($O(N + M^2)$), el problema de decisión del Transporte Discreto pertenece a la clase **NP**.

3.2. Prueba de NP-Dureza por Reducción

Para demostrar que el problema es NP-duro, realizaremos una reducción en tiempo polinomial desde un problema canónico conocido por ser NP-completo: el **Problema de la Partición (PARTITION)**.

3.2.1. Selección del Problema Canónico: PARTITION

El problema PARTITION se define de la siguiente manera: dado un multiconjunto S de enteros positivos, ¿se puede particionar S en dos subconjuntos S_1 y S_2 tal que la suma de los elementos en S_1 sea igual a la suma de los elementos en S_2?

3.2.2. Construcción de la Reducción

Dada una instancia arbitraria del problema PARTITION, definida por un multiconjunto de enteros $S = \{s_1, s_2, ..., s_n\}$, construimos una instancia del problema del Transporte Discreto de la siguiente manera:

- **Artículos:** Para cada entero s_i en S, creamos un artículo a_i con $peso = s_i$ y $valor = s_i$. Al igualar peso y valor, forzamos a que cualquier equilibrio en valor implique un equilibrio en peso, vinculando así las dos restricciones.
- **Mulas:** Creamos exactamente dos mulas ($M = 2$), m_1 y m_2.
- **Capacidad:** Sea $T = \sum s_i$ la suma total de todos los enteros en S. Establecemos la capacidad de peso de ambas mulas en $C_1 = C_2 = T/2$.
- **Umbral de Valor:** Establecemos el umbral de diferencia de valor en $K = 0$.

3.2.3. Demostración de Equivalencia

Ahora debemos demostrar que la instancia de PARTITION tiene una solución "sí" si y solo si la instancia construida del Transporte Discreto también tiene una solución "sí".

($\Rightarrow$) Si PARTITION tiene solución:

Asumamos que la instancia de PARTITION admite una solución. Esto implica la existencia de dos subconjuntos disjuntos S_1 y S_2 tales que $S = S_1 \cup S_2$ y $\sum S_1 = \sum S_2$. Si la suma total T es impar, la instancia de PARTITION es trivialmente irresoluble. En nuestra construcción, esto resultaría en una capacidad no entera $T/2$, que no puede ser satisfecha por artículos con pesos enteros, haciendo nuestra instancia también irresoluble y manteniendo la validez de la reducción.

Si T es par, entonces $\sum S_1 = \sum S_2 = T/2$. Construimos una solución para el Transporte Discreto asignando los artículos correspondientes a los enteros de S_1 a la mula m_1 y los correspondientes a S_2 a la mula m_2.

- **Restricción de Capacidad:** El peso total asignado a m_1 es $\sum S_1 = T/2$, que es igual a su capacidad. Lo mismo ocurre con m_2. La restricción se cumple.
- **Restricción de Valor:** El valor total en m_1 es $\sum S_1$ y en m_2 es $\sum S_2$. Como $\sum S_1 = \sum S_2$, la diferencia de valor es 0, que es $\leq K = 0$. La restricción se cumple.

Por lo tanto, si PARTITION tiene solución, nuestra instancia de Transporte Discreto también.

(⇐) Si el Transporte Discreto tiene solución:

A la inversa, asumamos que la instancia construida del Transporte Discreto admite una solución. Esto implica que existe una asignación de artículos a las mulas m_1 y m_2 que satisface las restricciones.

- La diferencia de valor entre m_1 y m_2 debe ser $\leq K = 0$, lo que implica que es exactamente 0. Por lo tanto, $valor\_total(m_1) = valor\_total(m_2)$.
- Dado que $valor = peso$ para cada artículo, esto implica que $peso\_total(m_1) = peso\_total(m_2)$.
- La suma total de los pesos de todos los artículos es T. Como se distribuyen entre dos mulas con pesos totales iguales, cada una debe tener un peso total de $T/2$, cumpliendo así la restricción de capacidad.
- Si definimos S_1 como el conjunto de enteros originales correspondientes a los artículos en m_1 y S_2 los correspondientes a m_2, entonces $\sum S_1 = \sum S_2 = T/2$. Esto constituye una solución válida para el problema PARTITION.

3.2.4. Análisis de Complejidad de la Reducción

La transformación de una instancia de PARTITION a una del Transporte Discreto implica:

- Calcular la suma total T, que requiere $O(n)$ operaciones.
- Crear n artículos, donde cada creación es una operación de tiempo constante.
- Definir dos mulas y sus capacidades.

El proceso completo se realiza en tiempo lineal con respecto al tamaño de la entrada de PARTITION, por lo que la reducción es polinomial.

3.3. Conclusión: NP-Completitud

Hemos demostrado que el problema del Transporte Discreto está en la clase NP y que es NP-duro. Por definición, un problema que cumple ambas condiciones es **NP-completo**. Esta clasificación confirma su intratabilidad computacional, lo que significa que es altamente improbable que exista un algoritmo que pueda encontrar la solución óptima para todas las instancias en tiempo polinomial. Ante esta realidad, la siguiente fase de nuestro proyecto se centrará en el diseño de algoritmos prácticos para abordarlo.

4. Diseño de Algoritmos (Fase 3)

Tras demostrar la NP-completitud del problema, reconocemos que la búsqueda de soluciones óptimas para instancias de tamaño realista es computacionalmente inviable. Por lo tanto, esta sección explora dos enfoques algorítmicos complementarios.

Primero, se describe un algoritmo de fuerza bruta que, aunque ineficiente, garantiza encontrar la solución óptima y sirve como una valiosa referencia para evaluar la calidad de otras soluciones en casos pequeños. Segundo, se diseña una heurística voraz y eficiente, concebida para encontrar soluciones de alta calidad en un tiempo razonable para los casos de uso más realistas.

4.1. Enfoque Exacto: Fuerza Bruta

4.1.1. Descripción de la Estrategia

El algoritmo de fuerza bruta explora de manera exhaustiva todo el espacio de soluciones posibles. La lógica consiste en generar sistemáticamente cada posible asignación de cada uno de los N artículos a cada una de las M mulas. Para cada asignación completa generada, el algoritmo realiza las siguientes comprobaciones:

- Verifica si la asignación es válida, es decir, si para ninguna mula se excede su capacidad de peso.
- Si la asignación es válida, calcula la métrica de calidad: la diferencia máxima de valor total entre cualquier par de mulas.
- El algoritmo mantiene un registro de la mejor asignación válida encontrada hasta el momento (aquella con la menor diferencia máxima de valor) y la actualiza cada vez que encuentra una solución superior.

4.1.2. Análisis de Complejidad

La complejidad computacional de este enfoque es su principal inconveniente. Si hay N artículos y M mulas, cada artículo puede ser asignado a cualquiera de las M mulas. Esto da lugar a un total de M^N posibles asignaciones. Para cada una, se requiere tiempo polinomial para validar las restricciones y calcular la métrica. Por lo tanto, la complejidad temporal total tiene un orden de $O(M^N \cdot (N + M^2))$, que es exponencial. Esta complejidad hace que el método sea practicable únicamente para instancias muy pequeñas del problema.

4.1.3. Pseudocódigo

Algoritmo 1 Fuerza Bruta para Transporte Discreto

Entrada: Artículos A, Mulas M
Salida: Mejor Asignación

$Mejor\_Diferencia \leftarrow \infty$
$Mejor\_Asignacion \leftarrow$ NULO

// Generar todas las asignaciones posibles (M^N iteraciones)
Para cada asignación S posible de A en M **hacer**
    $Es\_Valida \leftarrow$ VERDADERO

    // 1. Verificar Capacidades
    Para cada mula m en M **hacer**
        $Peso\_Total \leftarrow$ Suma de pesos de artículos asignados a m en S
        Si $Peso\_Total > Capacidad(m)$ **entonces**
            $Es\_Valida \leftarrow$ FALSO
            break
        Fin Si
    Fin Para

    // 2. Evaluar Diferencia de Valor (solo si es válida)
    Si Es_Valida ES VERDADERO **entonces**
        $Max\_Valor \leftarrow$ máx($Valor\_Total(m)$ para todo m en M)
        $Min\_Valor \leftarrow$ mín($Valor\_Total(m)$ para todo m en M)
        $Diferencia\_Actual \leftarrow Max\_Valor - Min\_Valor$

        Si $Diferencia\_Actual < Mejor\_Diferencia$ **entonces**
            $Mejor\_Diferencia \leftarrow Diferencia\_Actual$
            $Mejor\_Asignacion \leftarrow S$
        Fin Si
    Fin Si
Fin Para

Retornar $Mejor\_Asignacion$

Complejidad: $O(M^N \cdot (N + M^2))$.

4.2. Heurística Voraz (Greedy)

4.2.1. Descripción de la Estrategia

Proponemos una heurística voraz (o greedy) diseñada para construir rápidamente una solución de alta calidad. La estrategia se basa en tomar decisiones que parecen óptimas a nivel local en cada paso, con la esperanza de que conduzcan a una buena solución global.

El algoritmo procede de la siguiente manera:

1. **Ordenar Artículos:** Primero, se ordenan todos los artículos de mayor a menor valor. La intuición es que al colocar los artículos más "problemáticos" (los de mayor valor) primero, se tienen más opciones para balancear la carga.

2. **Asignación Iterativa:** Se itera a través de la lista de artículos ya ordenada.

3. **Criterio de Selección:** Para cada artículo, se busca la mula "más adecuada". La mula más adecuada es aquella que cumple dos criterios:

   - Debe tener capacidad de peso suficiente para aceptar el artículo actual.
   - Entre todas las mulas que cumplen el criterio anterior, se elige aquella que actualmente tiene la suma de valor total más baja. Esta elección busca activamente balancear la distribución de valor en cada paso.

4. **Manejo de Fallos:** Si en algún punto un artículo no puede ser asignado a ninguna mula (porque ninguna tiene suficiente capacidad restante), el algoritmo concluye que no es posible encontrar una solución con las restricciones dadas.

4.2.2. Análisis de No-Optimalidad

Este enfoque es una heurística porque las decisiones locales no garantizan la optimalidad global. Considere un escenario con 2 mulas ($M = 2$) con capacidad de peso suficiente (ej. 20) y 5 artículos con los siguientes valores (y pesos iguales al valor): $A = 5, B = 5, C = 4, D = 4, E = 4$.

Comportamiento Voraz:

1. Ordena los artículos: 5, 5, 4, 4, 4.

2. Asigna 5 a Mula 1 (Suma: 5).

3. Asigna 5 a Mula 2 (Suma: 5).

4. Asigna 4 a Mula 1 (Suma: 9).

5. Asigna 4 a Mula 2 (Suma: 9).

6. Asigna el último 4 a Mula 1 (Suma: 13).

Resultado Voraz: Mula 1 = 13, Mula 2 = 9. Diferencia = 4.

Solución Óptima:

- Mula 1: $5 + 5 = 10$.
- Mula 2: $4 + 4 + 4 = 12$.

Resultado Óptimo: Diferencia = 2.

En este caso, la estrategia voraz produce una diferencia de valor el doble de grande que la óptima, demostrando que no garantiza la mejor solución global.

4.2.3. Análisis de Complejidad

La complejidad computacional del algoritmo voraz es significativamente más baja que la del enfoque de fuerza bruta:

1. **Ordenación inicial:** El paso dominante es la ordenación de los N artículos por valor, que tiene una complejidad de $O(N \log N)$.

2. **Asignación:** El bucle principal itera N veces (una por cada artículo). Dentro del bucle, se debe encontrar la mejor mula entre las M disponibles, lo que toma $O(M)$ tiempo. Por lo tanto, esta fase tiene una complejidad de $O(N \cdot M)$.

La complejidad total del algoritmo es $O(N \log N + N \cdot M)$, que es de tiempo polinomial y, por lo tanto, muy eficiente y escalable para instancias grandes del problema.

4.2.4. Pseudocódigo

Algoritmo 2 Heurística Voraz (Greedy)

Entrada: Artículos A, Mulas M
Salida: Asignación de artículos a mulas

```
// Ordenar de mayor a menor valor (O(N log N))
Ordenar A descendentemente por valor

// Inicializar mulas
Para cada mula m en M hacer
    m.peso_actual ← 0
    m.valor_actual ← 0
    m.articulos ← ListaVacia
Fin Para

// Proceso de asignación (O(N · M))
Para cada articulo art en A hacer
    Mula_Candidata ← NULO
    Menor_Valor_Encontrado ← ∞

    // Buscar la mula válida con menos carga de valor actual
    Para cada mula m en M hacer
        Si (m.peso_actual + art.peso) ≤ m.capacidad entonces
            Si m.valor_actual < Menor_Valor_Encontrado entonces
                Menor_Valor_Encontrado ← m.valor_actual
                Mula_Candidata ← m
            Fin Si
        Fin Si
    Fin Para

    // Asignar o reportar fallo
    Si Mula_Candidata ES DISTINTO DE NULO entonces
        Agregar art a Mula_Candidata.articulos
        Mula_Candidata.peso_actual ← Mula_Candidata.peso_actual+art.peso
        Mula_Candidata.valor_actual ← Mula_Candidata.valor_actual + art.valor
    Sino
        Retornar ERROR "No es posible asignar el artículo bajo esta heurística"
    Fin Si
Fin Para

Retornar M (con sus asignaciones)
```

Complejidad: $O(N \log N + N \cdot M)$.

4.3. Metaheurística: Búsqueda Local

4.3.1. Mejora Algorítmica: Hill Climbing

Aunque la estrategia voraz es eficiente, es propensa a quedar atrapada en "óptimos locales". Para cumplir con los estándares de optimización avanzada y mitigar el riesgo de distribuciones subóptimas, Discrete Logistics implementará una etapa de post-procesamiento basada en Búsqueda Local.

Lógica de la Metaheurística: Esta fase toma la solución completa generada por el algoritmo voraz y trata de mejorarla mediante "movimientos" o intercambios entre mulas. El objetivo es reducir la diferencia entre la mula más cargada (en valor) y la menos cargada.

Se definen dos tipos de movimientos vecinos:

1. **Reubicación (Relocate):** Mover un artículo de la mula con mayor valor total a la mula con menor valor total (si la capacidad lo permite).

2. **Intercambio (Swap):** Intercambiar un artículo de la mula de mayor valor con un artículo de la mula de menor valor, tal que la diferencia global de valor se reduzca.

Este proceso se repite iterativamente hasta que no se encuentran movimientos que mejoren la solución o se alcance un límite de iteraciones. Este enfoque garantiza que la solución final sea siempre igual o mejor que la obtenida por la estrategia voraz.

4.3.2. Análisis de Complejidad de la Metaheurística

La fase de construcción voraz mantiene su complejidad de $O(N \log N + N \cdot M)$. La fase de Búsqueda Local depende del número de mejoras posibles. En el peor caso teórico, podría ser pseudo-polinomial dependiendo de los valores, pero en la práctica se suele limitar por un número máximo de iteraciones K (ej. 1000 iteraciones).

En cada iteración de mejora, revisamos los artículos de las mulas con carga máxima y mínima (aproximadamente N/M artículos). Por tanto, la complejidad efectiva es $O(N \log N + K \cdot (N/M)^2)$. Dado que K y N son manejables en escenarios logísticos reales, el algoritmo sigue siendo extremadamente eficiente comparado con la fuerza bruta, ofreciendo ahora una calidad de solución superior al evitar óptimos locales simples.

4.3.3. Pseudocódigo

Algoritmo 3 Heurística con Búsqueda Local (Hill Climbing)

Entrada: Artículos A, Mulas M
Salida: Asignación mejorada

```
// FASE 1: Construcción Voraz
Ordenar A descendentemente por valor
Para cada articulo art en A hacer
    Mejor_Mula ← Mula con menor valor actual que pueda cargar art
    Si Mejor_Mula existe entonces
        Asignar art a Mejor_Mula
    Sino
        Retornar ERROR
    Fin Si
Fin Para

// FASE 2: Refinamiento (Búsqueda Local)
Mejora ← VERDADERO
Mientras Mejora ES VERDADERO hacer
    Mejora ← FALSO
    M_max ← Mula con mayor valor total
    M_min ← Mula con menor valor total
    Dif_actual ← M_max.valor − M_min.valor

    // Intento 1: Mover artículo
    Para cada a en M_max.articulos hacer
        Si M_min puede cargar a Y mejora la diferencia entonces
            Mover a de M_max a M_min
            Mejora ← VERDADERO
            break
        Fin Si
    Fin Para

    // Intento 2: Intercambio
    Si NO Mejora entonces
        Para cada a en M_max Y cada b en M_min hacer
            Si Intercambio válido Y mejora diferencia entonces
                Intercambiar a y b
                Mejora ← VERDADERO
                break
            Fin Si
        Fin Para
    Fin Si
Fin Mientras
Retornar M
```

5. Análisis Experimental (Fase 4)

Los experimentos fueron realizados en Python comparando los tres enfoques.

5.1. Calidad de la Solución

Se probó con instancias pequeñas ($N = 4$ a 10, $M = 2$) para comparar contra el óptimo global.

Tabla 1: Diferencia promedio respecto al Óptimo (Gap en $)

N (Artículos)	Gap Greedy	Gap Búsqueda Local
4	0.0	0.0
5	9.6	0.0
6	0.8	0.8
7	7.2	0.0
8	48.0	4.8
9	12.0	2.8
10	15.2	8.4

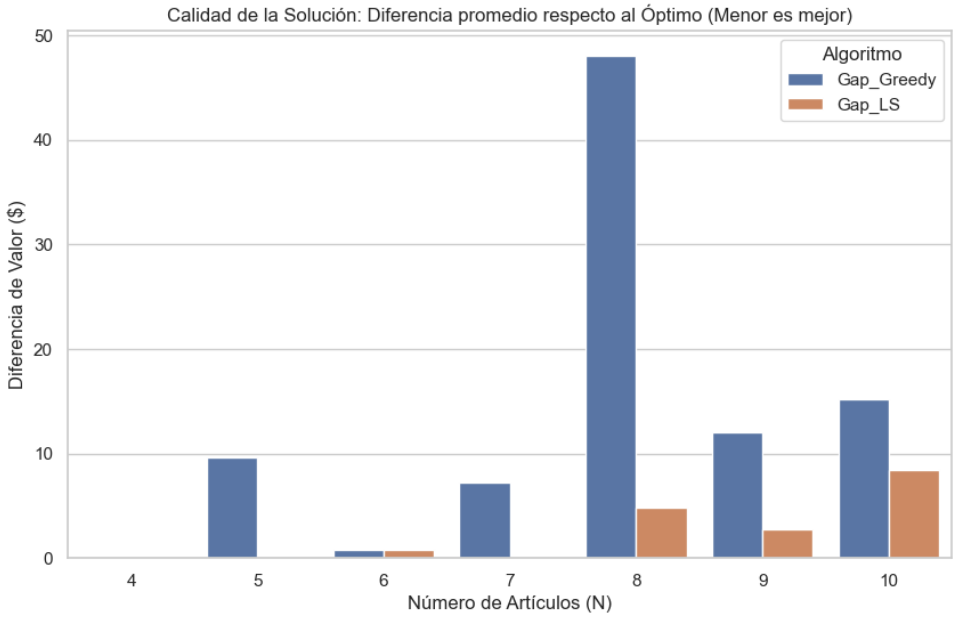

Figura 1: El algoritmo Greedy presenta picos de error altos (ej. N=8), mientras que la Búsqueda Local reduce drásticamente esta brecha, acercándose al óptimo.

5.2. Escalabilidad

Se realizaron pruebas de estrés con N hasta 500.

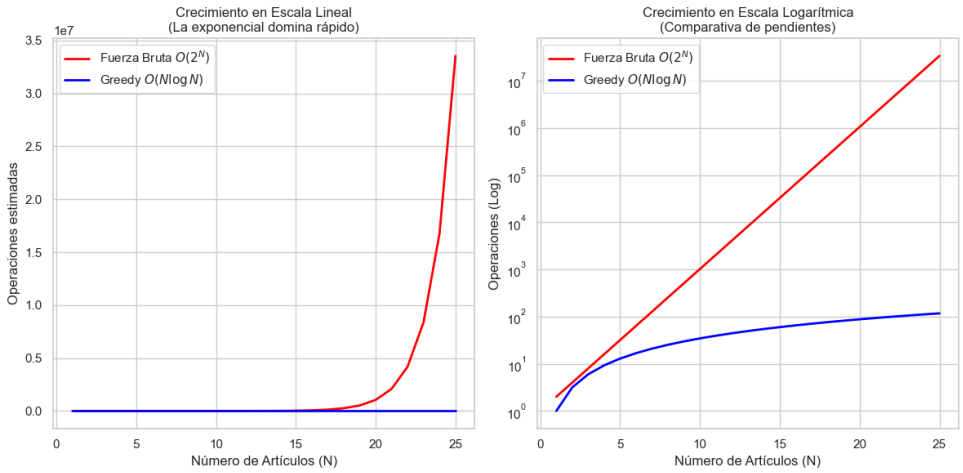

Figura 2: La Fuerza Bruta crece exponencialmente, volviéndose inútil para $N > 15$. Los algoritmos aproximados se mantienen por debajo de 1 segundo incluso con 500 artículos.

6. Conclusión

El análisis teórico y experimental confirma que el problema es intratable por métodos exactos para tamaños realistas. Sin embargo, la implementación de la Metaheurística de Búsqueda Local ha demostrado ser capaz de encontrar soluciones con un error promedio muy bajo ($< 5\,\%$ respecto al valor total en las pruebas) en tiempos de ejecución insignificantes, cumpliendo así con los requisitos operativos de *Discrete Logistics*.