# Optimización del Llenado Manual de Contenedores con Paquetes Heterogéneos

## Máster Universitario en Estadística Computacional y Ciencia de Datos para la Toma de Decisiones

Jose Gustavo Quilca Vilcapoma
Tutor: Javier Alcaraz Soria

Instituto Centro de Investigación Operativa
Universidad Miguel Hernández

17 Junio 2024



# Introducción

- El problema del llenado de contenedores (CLP) es un problema clásico en la optimización combinatoria.
- Consiste en encontrar la mejor manera de llenar un contenedor con cajas de diferentes tamaños y pesos, optimizando la utilización del espacio.
- Este problema tiene aplicaciones en logística, transporte, almacenamiento, entre otros.

# Restricciones

- Restricciones Básicas
  - No superposición de paquetes.
  - No superar el peso máximo del contenedor.
  - Colocación dentro de los límites del contenedor.
- Restricciones Prácticas
  - Estabilidad de los paquetes.
  - Centro de gravedad del contenedor.
  - Prioridad de carga.
  - Restricción de contigüidad de tipos.

# Métodos de Solución

- **Métodos exactos**:
  - Garantizan la solución óptima, pero son computacionalmente costosos.
- **Métodos heurísticos**:
  - Proporcionan soluciones de buena calidad en un tiempo razonable.
- **Métodos metaheurísticos**:
  - Estrategias flexibles de alto nivel para desarrollar algoritmos que resuelven problemas específicos.

# Contexto del Problema

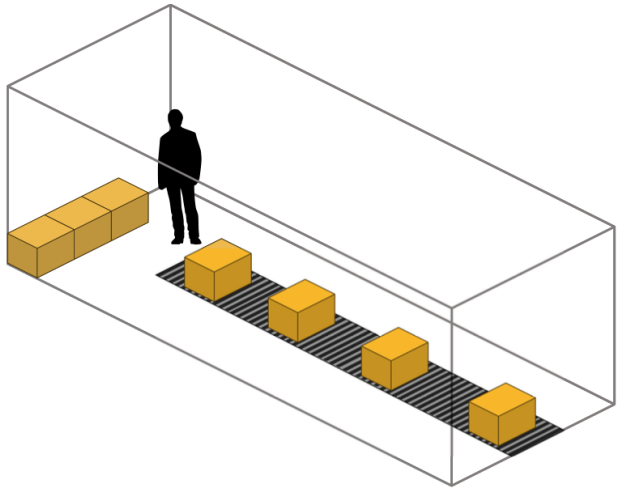
- Empresa en el sector de envío de paquetes en contenedores marítimos.
- No tiene control sobre las medidas de los paquetes.
- Cuenta con una cantidad máxima de paquetes de cada tipo.
- Envía un solo contenedor a la vez.
- Usan procedimientos establecidos de llenado manual.

# Definición del Problema

- Maximizar el beneficio de la carga en un contenedor, con los paquetes disponibles.
- Los paquetes apilados deben mantener su estabilidad durante la carga.
- Todos los paquetes del mismo tipo son cargados de forma contigua.
- Los paquetes pueden ser rotados en una sola dirección.

# Definición Formal

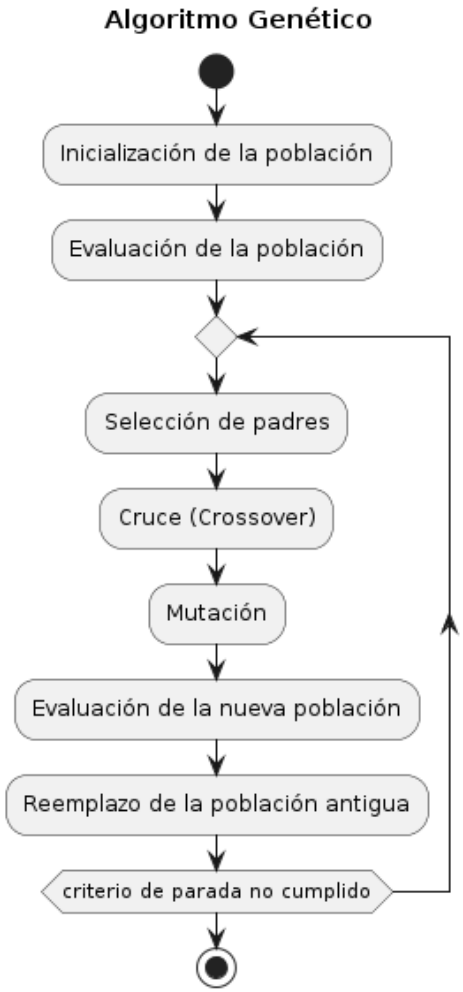
Optimizar la carga de un contenedor, determinando el **orden**, la **cantidad** y **rotación** de los tipos de paquetes, cumpliendo con las restricciones prácticas del **llenado manual**. El objetivo principal es maximizar el beneficio total de la carga junto a la utilización del espacio del contenedor.



# Algoritmo Genético

## Es una técnica de optimización inspirada en la evolución natural:

- Inicia con una población aleatoria de soluciones.
- Evalúa su calidad con una función de aptitud.
- Selecciona las mejores para reproducirse mediante cruces y mutaciones.
- Reemplaza la población anterior con nuevas generaciones.
- Repite el ciclo hasta alcanzar una solución óptima o cumplir un criterio de parada.



# Algoritmo Genético

## Codificación de las Soluciones:

- Uso de 3 listas correlacionadas para representar el orden de llenado.
- Ayuda a garantizar factibilidad.
- Sirve de guía para el operario.

| | | | | |
|---|---|---|---|---|
| Tipo | 2 | 1 | 4 | 3 |
| Rotación | 0 | 1 | 1 | 0 |
| Cantidad | 27 | 10 | 18 | 13 |

Ejemplo de codificación

# Algoritmo Genético

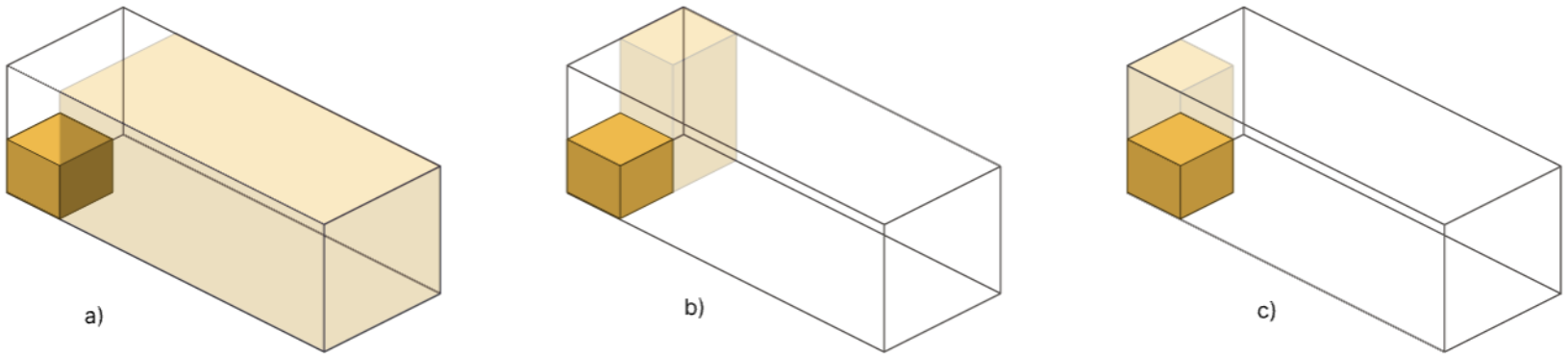
## Función de evaluación:

- Calcula el beneficio total de la carga.
- Es necesario el desarrollo de un algoritmo que **imita el procedimiento de llenado manual**.
- Verificar la factibilidad de la soluciones.

# Algoritmo Genético: Función de evaluación

## Algoritmo de llenado manual:

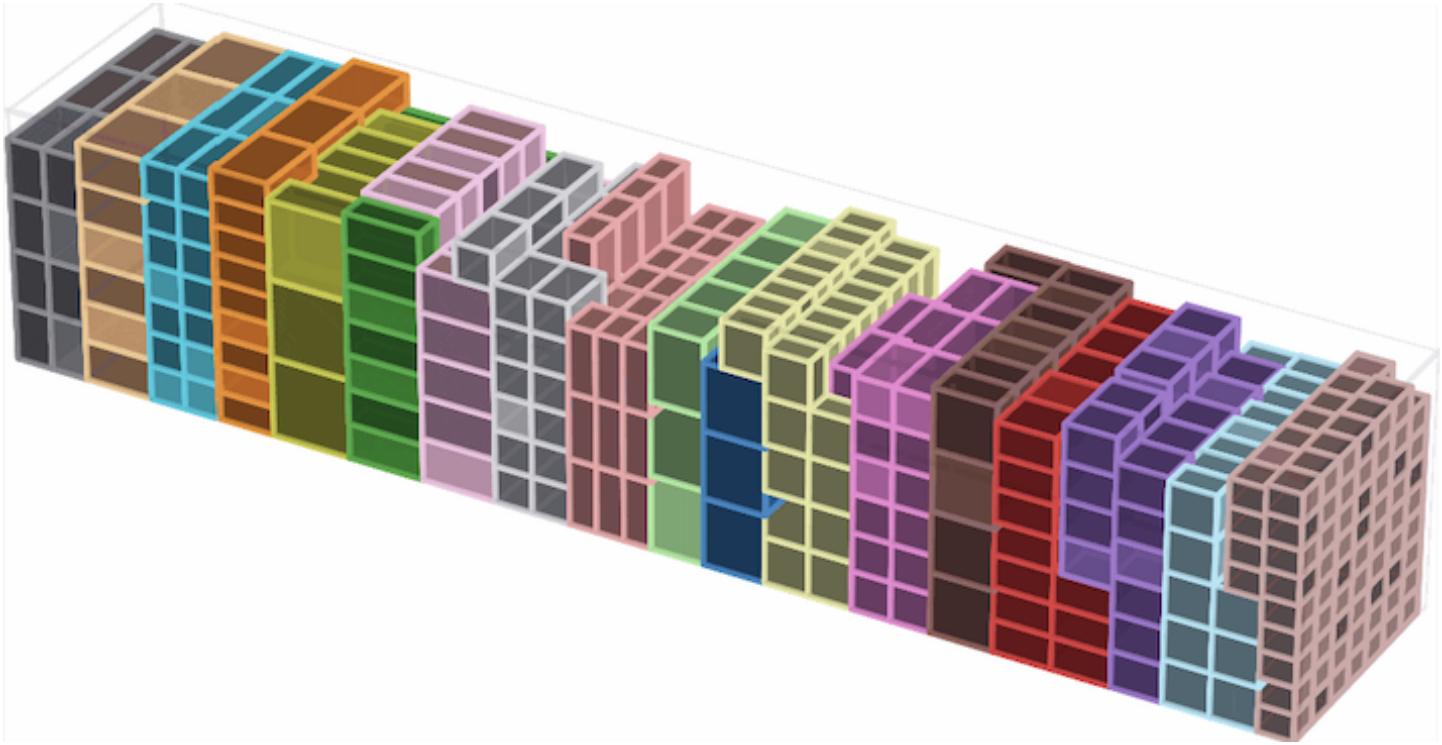
Basado en el método Deepest Bottom Left with Fill (DBLF) propuesto por Karabulut 2005.




División del espacio restante en el contenedor luego de colocar un paquete

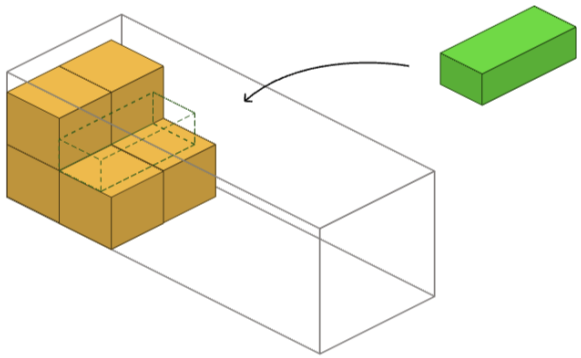
# Llenado Manual

## Ejemplo de llenado manual



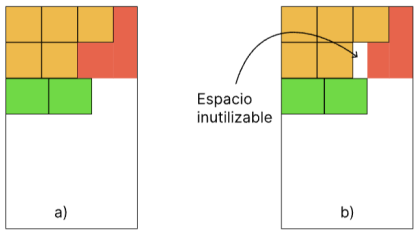
Ejemplo del llenado de un contenedor con 20 tipos de paquetes

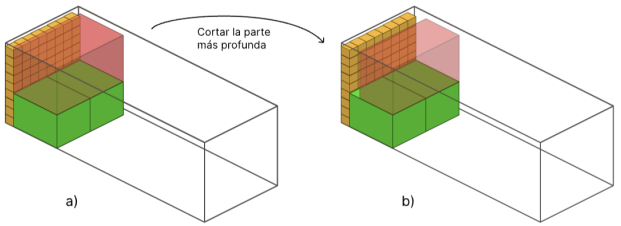
# Algoritmo Genético: Función de evaluación



## Adaptación del Algoritmo DBLF:

- Unión de subespacios
- Eliminación de subespacios inaccesibles
- Eliminación de subespacios profundos



# Algoritmo Genético: Población Inicial y Selección

## Generación aleatoria de la población inicial

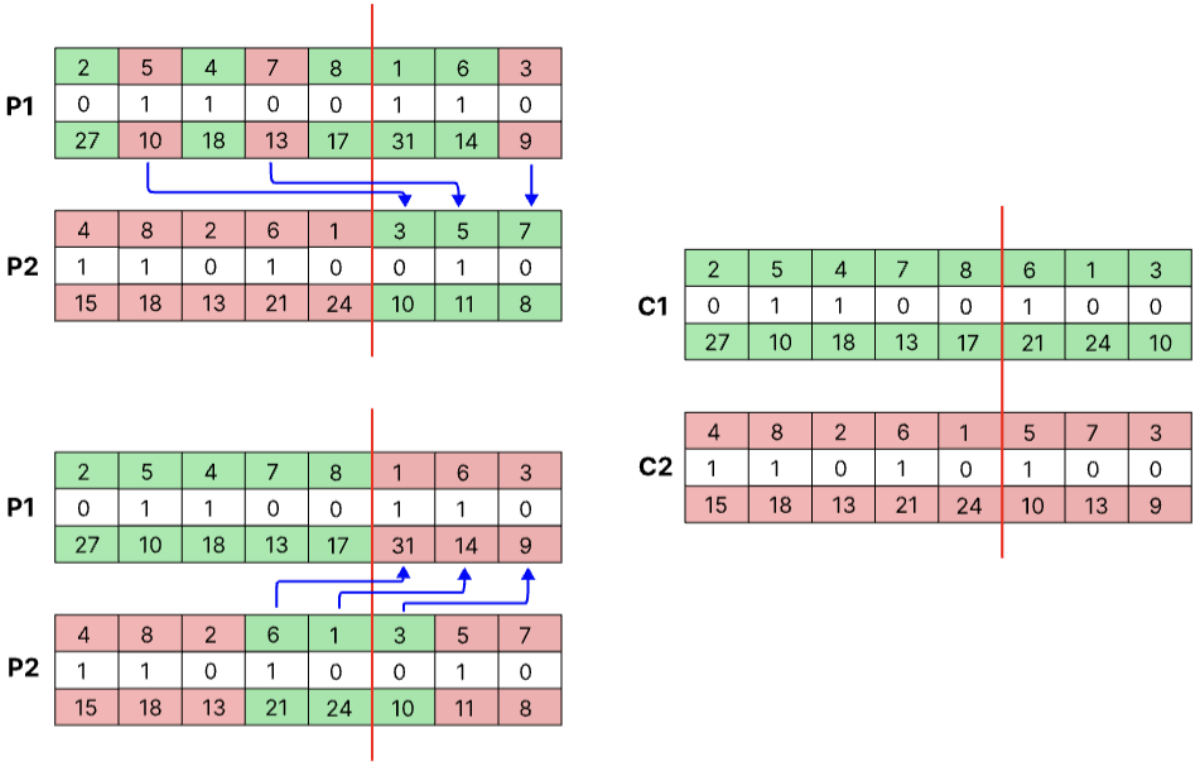
| | | | | |
|---|---|---|---|---|
| Tipo | 2 | 1 | 4 | 3 |
| Rotación | 0 | 1 | 1 | 0 |
| Cantidad | 27 | 10 | 18 | 13 |

Ejemplo de un individuo de la población

## Método de torneo para la selección

# Algoritmo Genético: Cruce

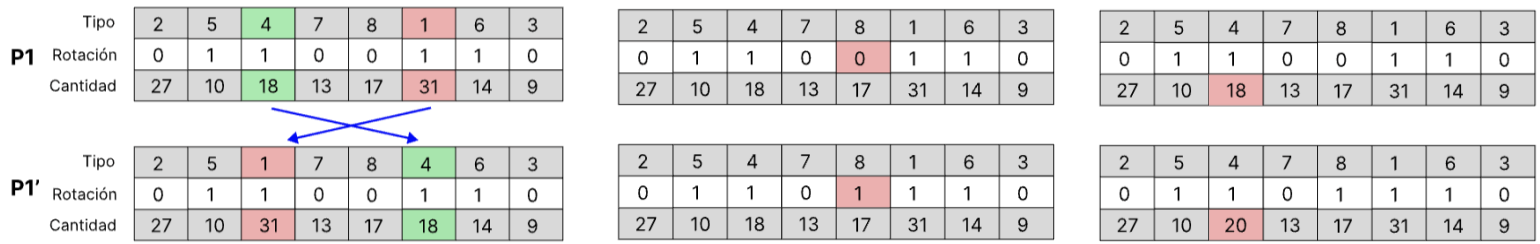
## Adaptación del operador de cruce de un punto




Método de cruce que evita generar soluciones no factibles

# Algoritmo Genético: Mutación

## Adaptación del operador de mutación de tres tipos

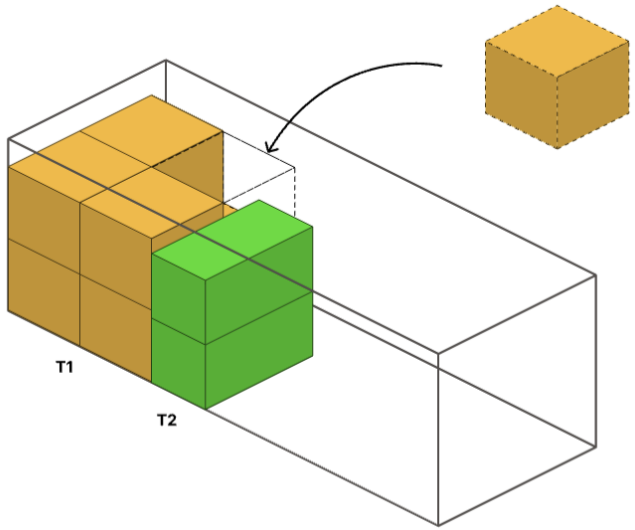



Métodos de mutación para el orden, la rotación y la cantidad por tipo

# Algoritmo de Mejora

## Mejoras del Algoritmo DBLF:

- Ayuda a completar espacios vacíos si aún hay paquetes disponibles.
- Aplicarla durante el llenado podría empeorar una solución.
- Aplicarla luego del llenado garantiza que la solución no sea peor.
- Agregan tiempo adicional de cómputo.



# Estudio Computacional

## Generación de Datos de Prueba

- Un contenedor con dimensiones fijas $12010 \times 2330 \times 2380mm$.
- Instancias con 5, 10, 20, 30, 40 y 50 tipos de cajas.
- Tamaños de cajas aleatorias entre 250 y 750mm
- Valores aleatorios entre 10 y 100

# Estudio Computacional

## Configuración del Algoritmo

- Algoritmos implementados en Python 3.10.
- $P_{cross} = 0.8$
- $P_{mut} = 0.05$.
- Población inicial de 100 individuos, generados aleatoriamente.

# Estudio Computacional

## Variantes del algoritmo

Comparar las mejoras propuestas

- **M0**: Sin mejora del algoritmo en el llenado.
- **M1**: Mejora de llenado adicional inmediato.
- **M2**: Mejora de llenado adicional al final.
- **M3**: Mejora similar a M2, solo aplicada a la mitad de la población.
- **M4**: Mejora similar a M2, solo aplicada al mejor individuo de la población.

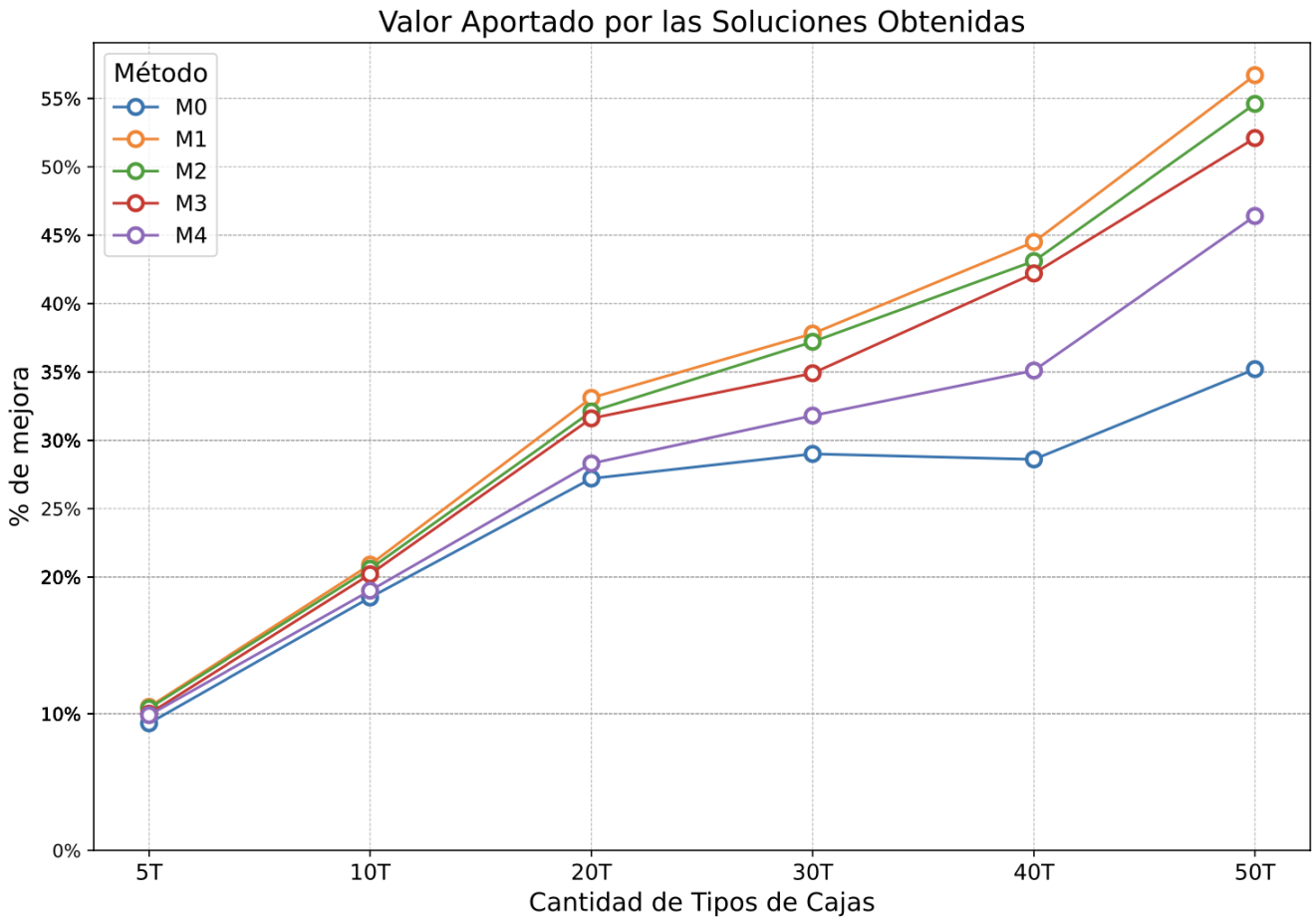
# Estudio Computacional

## Configuración del Experimento

- 25 instancias para cada tipo de caja (5T, 10T, 20T, 30T, 40T y 50T), total 150 instancias.
- Tiempo fijo de 5 minutos para cada problema.
- Tiempo de ejecución total de 62.5 horas (2.6 días).
- Se recolectaron datos en csv para su análisis.

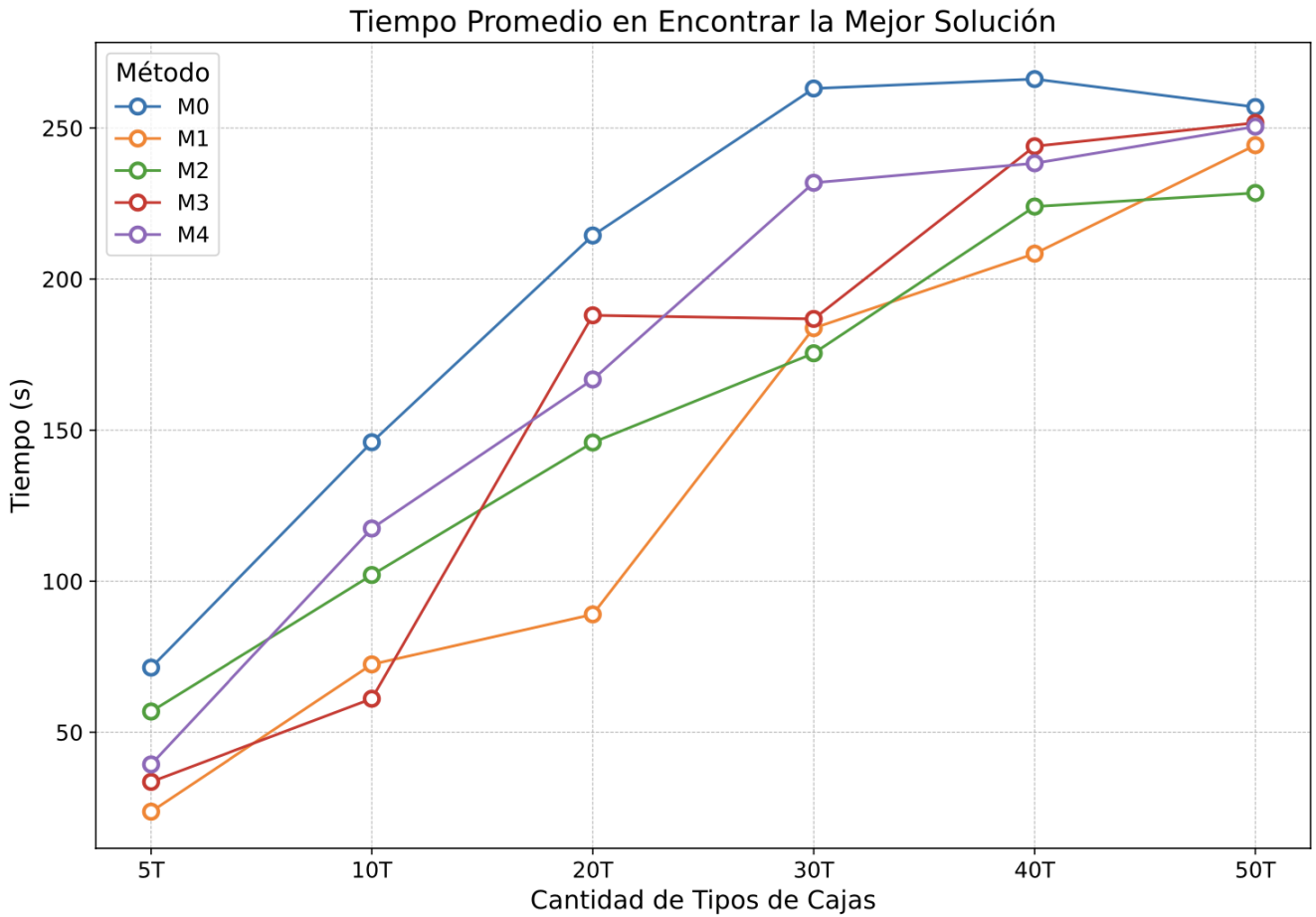
# Estudio Computacional: Resultados

## Beneficio aportado por las soluciones



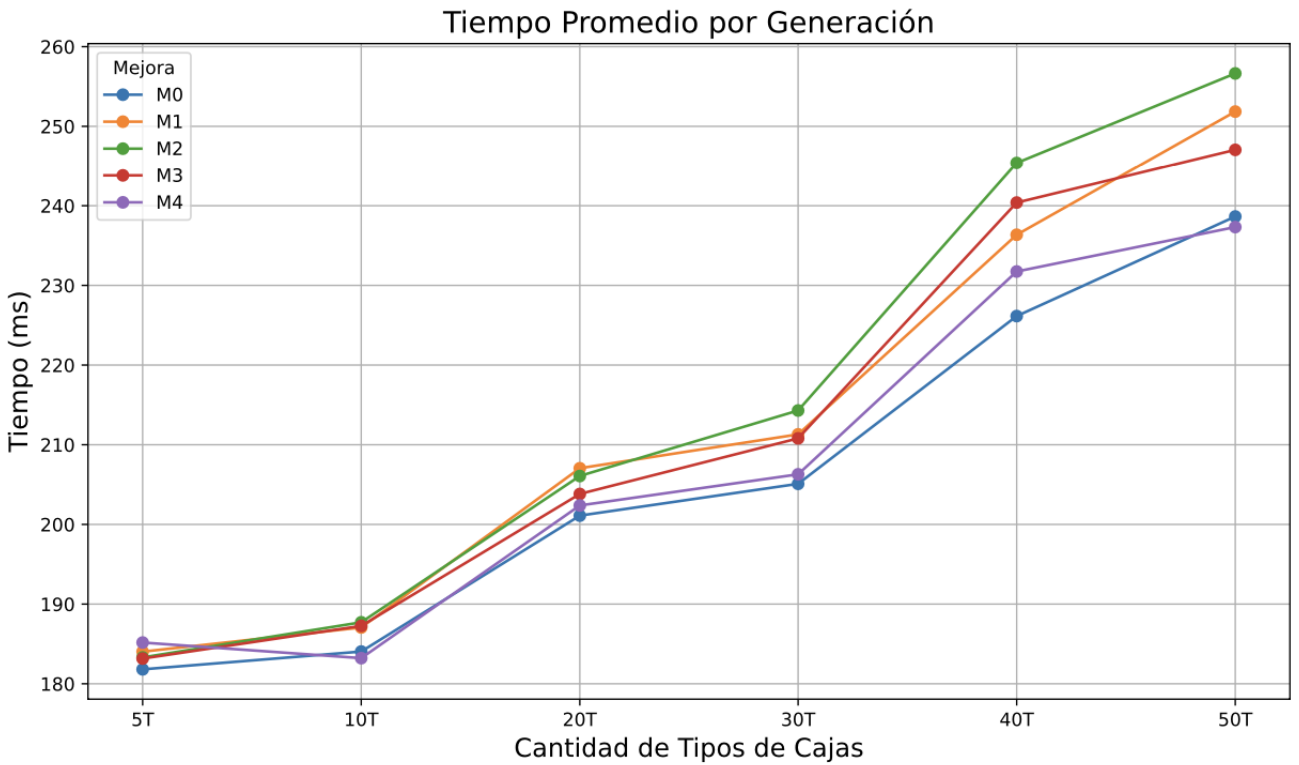
# Estudio Computacional: Resultados

## Tiempo promedio en encontrar la mejor solución



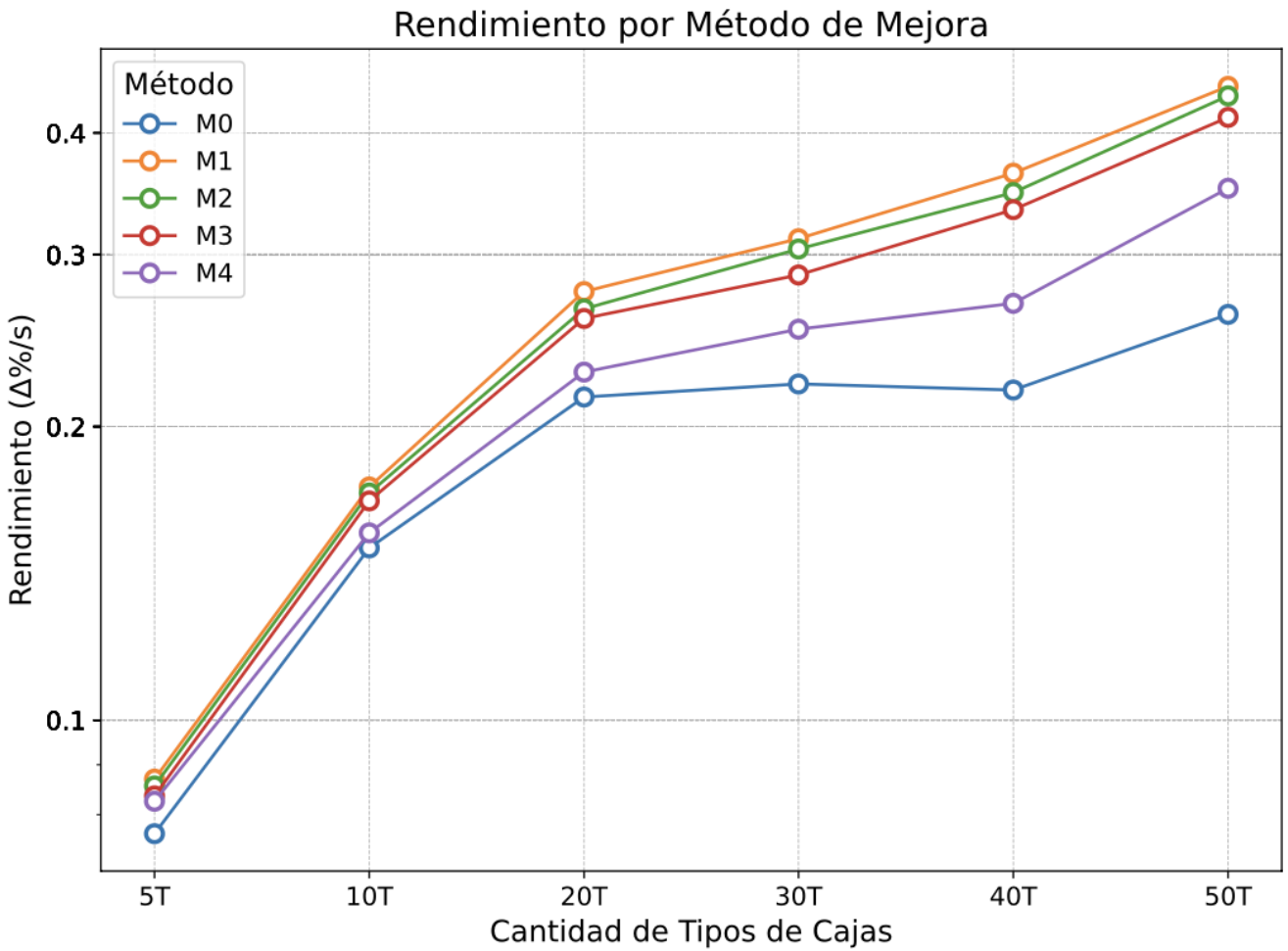
# Estudio Computacional: Resultados

## Tiempo promedio por generación



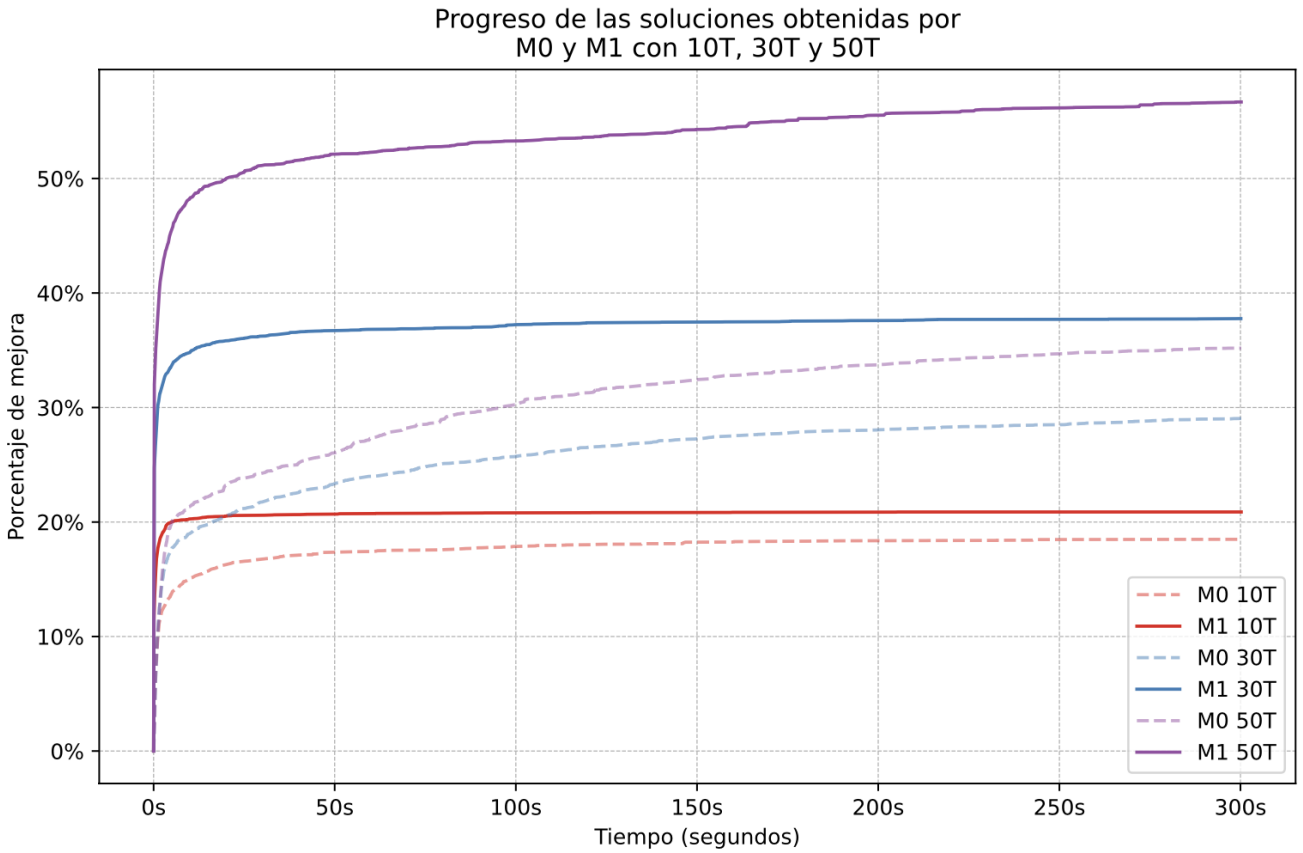
# Estudio Computacional: Resultados

## Rendimiento por método de mejora



# Estudio Computacional: Resultados

## Progreso de las soluciones del mejor método elegido: M1



# Conclusiones

- El algoritmo ha demostrado ser un método eficiente para determinar la carga del contenedor y la forma de llenarlo.
- Los métodos de mejoras consiguen reducir el tiempo necesario para encontrar la mejor solución.
- También consiguen mejorar la calidad de las soluciones.
- Elevado número de tipos de cajas necesitan más tiempo para converger.

# Trabajos Futuros

- Diseñar otros tipos de operadores de cruce y mutación.
- Considerar otras restricciones prácticas.
- Probar el algoritmo con datos reales.
- Incluir e algoritmo en un DSS utilizado en tiempo real.

*Gracias*