



UNIVERSIDAD DE CONCEPCIÓN  
FACULTAD DE INGENIERÍA  
DEPARTAMENTO DE INGENIERÍA CIVIL INDUSTRIAL



# UN ALGORITMO BASADO EN MACHINE LEARNING PARA EL PROBLEMA POLINOMIAL ROBUSTO DE LA MOCHILA

**Por: José Ignacio González Cortés**

Memoria de título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Industrial

Diciembre 2023

**Profesor Guía: Carlos Contreras Bolton**

© 2023, José Ignacio González Cortés

Ninguna parte de esta memoria puede reproducirse o transmitirse bajo ninguna forma o por ningún medio o procedimiento, sin permiso por escrito del autor.

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento



## AGRADECIMIENTOS

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

## Resumen

*Keywords* – Knapsack Problem

# Abstract

*Keywords* – Knapsack Problem

# Índice general

<b>AGRADECIMIENTOS</b>	<b>I</b>
<b>Resumen</b>	<b>II</b>
<b>Abstract</b>	<b>III</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Antecedentes generales . . . . .	1
1.2. Objetivos . . . . .	2
1.3. Estructura del documento . . . . .	3
<b>2. Problema Polinomial Robusto de la mochila</b>	<b>4</b>
2.1. Descripción del problema . . . . .	4
2.2. Modelo matemático . . . . .	5
2.2.1. Parametros . . . . .	5
2.2.2. Variables de decision: . . . . .	6
2.2.3. Función objetivo . . . . .	6
2.2.4. Restricciones . . . . .	6
2.3. Revisión de literatura . . . . .	7
<b>3. Metodología</b>	<b>9</b>
3.1. Algoritmo propuesto . . . . .	9
3.2. Clasificador . . . . .	10
3.2.1. Entrenamiento . . . . .	12
3.3. Reducción de instancias . . . . .	15
<b>4. Resultados Experimentales</b>	<b>17</b>
4.1. Instancias . . . . .	17
4.2. Comparaciones . . . . .	17
<b>5. Conclusiones</b>	<b>21</b>
<b>Referencias</b>	<b>22</b>
<b>Apéndices</b>	<b>24</b>

<b>A. Material</b>	<b>24</b>
A.1. Rendimiento de métodos en distintos tamaños de entrada . . . . .	24
<b>B. Ecuaciones</b>	<b>32</b>
B.1. Features Clasificador . . . . .	32



# Índice de Tablas

1.	Beneficios y costos de los ítems . . . . .	5
2.	Beneficios por sinergias . . . . .	5
3.	Comparación general por número de elementos . . . . .	20
4.	Tabla completa de resultados . . . . .	31

# Índice de Figuras

1.	Estructura general de la red . . . . .	11
2.	Tiempos de ejecución solver exacto . . . . .	18
3.	Gap % de todos los metodos presentados . . . . .	19
4.	TimeGap de los métodos usados . . . . .	19
5.	Rendimiento BaldoML . . . . .	24
6.	Rendimiento BaldoGA . . . . .	25
7.	Rendimiento Algorimo propuesto ( $\tau = 0.05$ ) . . . . .	25
8.	Rendimiento Algorimo propuesto ( $\tau = 0.1$ ) . . . . .	26
9.	Rendimiento Algorimo propuesto ( $\tau = 0.15$ ) . . . . .	26
10.	Rendimiento Algorimo propuesto ( $\tau = 0.2$ ) . . . . .	27
11.	Rendimiento Algorimo propuesto ( $\tau = 0.25$ ) . . . . .	27
12.	Rendimiento Algorimo propuesto ( $\tau = 0.3$ ) . . . . .	28
13.	Rendimiento Algorimo propuesto ( $\tau = 0.35$ ) . . . . .	28
14.	Rendimiento Algorimo propuesto ( $\tau = 0.4$ ) . . . . .	29
15.	Rendimiento Algorimo propuesto ( $\tau = 0.45$ ) . . . . .	29
16.	Rendimiento Algorimo propuesto ( $\tau = 0.5$ ) . . . . .	30

# Capítulo 1

## Introducción

En este capítulo se presentan los antecedentes generales acerca del Problema polinomial robusto de la mochila, las variantes de las cuales se deriva y metodologías de solución asociadas a estas. Los objetivos generales y específicos de esta memoria y la estructura del documento.

### 1.1. Antecedentes generales

El problema de la mochila (KP, por sus siglas en inglés, Knapsack Problem) es un problema clásico de la investigación de operaciones. Generalmente, el KP modela la necesidad de elegir un conjunto de elementos con costos y beneficios individuales, con una restricción de capacidad máxima, con el fin de maximizar el beneficio. El KP ha sido ampliamente estudiado por su estructura sencilla, y también debido a que muchos otros problemas de optimización más complejos lo tienen como subproblema ([Martello and Toth, 1990](#)).

El KP tiene muchas variantes, una ellas es la versión robusta, RKP (por sus siglas en inglés, Robust Knapsack Problem). Este fue formulado originalmente por [Eilon \(1987\)](#) para resolver problemas de asignación de presupuesto con aplicaciones reales, muchos de los parámetros del problema están asociados a incertidumbre. El RKP fue planteado para encontrar soluciones que sean factibles para todas las posibles variaciones en los costos de elementos ([Monaci et al., 2013](#)).

Otra variante es la versión polinomial de la mochila (PKP, por sus siglas en inglés, Polynomial Knapsack Problem) que incluye el concepto de sinergias, es decir, que la elección de una o más alternativas específicas otorga un beneficio o costo extra según estas

relaciones. El PKP sirve para modelar sistemas cuyas alternativas presentan conflictos entre ellas, o que cooperan para generar mayor beneficio (Baldo et al., 2023). Así, de este problema surge el polinomial robusto (PRKP, por sus siglas en inglés, Polynomial Robust Knapsack Problem). El PRKP toma en cuenta parámetros inciertos y sinergias polinomiales para modelar problemas de selección de alternativas, que se perjudican o benefician entre sí y además muestran comportamiento estocástico.

Debido a la complejidad espacial del PRKP, se han explorado aplicaciones del problema cuadrático de la mochila (QKP, por sus siglas en inglés, Quadratic Knapsack Problem) (Gallo et al., 1980) y el problema cúbico de la mochila (CKP, por sus siglas en inglés, Cubic Knapsack Problem) (Forrester and Waddell, 2022). El QKP presenta sinergias entre dos elementos y ha demostrado ser útil en un gran espectro de aplicaciones como posicionamiento satelital (Witzgall, 1975), localizaciones de centros de transporte como aeropuertos, ferrocarriles o terminales de carga (Rhys, 1970). El CKP es extendido desde el QKP y considera sinergias hasta con tres elementos. Además, posee aplicaciones como en el problema de satisfacción Max 3-SAT (Kofler et al., 2014), el problema de selección (Gallo et al., 1989), el problema de alineación de redes (Mohammadi et al., 2017), y la detección y tratamiento de enfermedades de transmisión sexual (Zhao, 2008).

Este trabajo propone la exploración de técnicas de machine learning para resolver el PRKP, encontrar una metodología competitiva con aquellas planteadas en la literatura y obtener resultados comparables.

## 1.2. Objetivos

### Objetivo general

Implementar una heurística basada en machine learning para resolver el PRKP.

### Objetivos específicos

- Revisar la literatura relacionada con problemas de la mochila similares y metodologías aplicables.
- Diseñar una heurística basada en machine learning para el PRKP.
- Implementar la heurística propuesta basada en machine learning.
- Evaluar los resultados y comparar el rendimiento con las metodologías expuestas

---

anteriormente desde la literatura.

## 1.3. Estructura del documento

El documento consta de 6 capítulos, en los cuales se discutirá, el modelo matemático representando el problema (2), la metodología propuesta para resolverlo (3), un análisis comparativo de los resultados experimentales (4), y finalmente una revisión y discusión para concluir el documento. (5).

## Capítulo 2

# Problema Polinomial Robusto de la mochila

Este capítulo comprende la definición formal del problema junto al modelo matemático de programación entera. Además, explora literatura asociada a técnicas y metodologías empleadas para la resolución de otros problemas de la mochila que pueden ser extendidos al PRKP.

### 2.1. Descripción del problema

El PRKP se define formalmente como un conjunto de elementos  $I = \{1, 2, \dots, N\}$  donde estos tienen un beneficio asociado  $P_i \in \mathbb{R} : i \in I$ . Además, los elementos poseen un coste de comportamiento estocástico que puede variar de forma continua entre una cota inferior y una superior,  $C_i \in [LC_i, UC_i]$ , donde  $LC_i$  es el costo base, y  $UC_i$  el costo máximo. De forma aleatoria solo algunos elementos tienen comportamiento estocástico, aquellos que no, cumplen que  $C_i = LC_i$ . El parámetro que define cuantos elementos de  $I$  pueden tener comportamiento estocástico es  $\Gamma \leq N$ , es decir  $|\{i : C_i > LC_i\}| \leq \Gamma$ . Además existe un presupuesto máximo  $W$  para los costes.

El conjunto de sinergias polinomiales se define como  $S = \{A : A \subseteq I \wedge |A| > 1\}$ . Para cada subconjunto de elementos de  $I$ , existe un beneficio asociado  $PS_A$ .

El objetivo del problema es encontrar un subconjunto de elementos  $X \subseteq I$  que maximice el beneficio total de los elementos de  $X$  sumado a los beneficios de sinergias, que aplican cuando  $A \subseteq X$ , y que además, para cualquier variación estocástica en costes de los

elementos.

## Ejemplo de PRKP

$$I = \{1, 2, 3\}$$

$$W = 10.0$$

$$\Gamma = 2$$

$i$	1	2	3
$PS_i$	3.0	6.0	10.0
$LC_i$	2.0	3.0	4.0
$UC_i$	5.0	4.0	7.0

**Tabla 1:** Beneficios y costos de los ítems

$A$	$PS_A$
$\{1, 2\}$	1.0
$\{1, 3\}$	2.0
$\{2, 3\}$	-1.0
$\{1, 2, 3\}$	3.0

**Tabla 2:** Beneficios por sinergias

## 2.2. Modelo matemático

Baldo et al. (2023) describe la formulación del modelo matemático linealizado del PRKP compatible con solvers exactos de programación lineal. El modelo se describe a continuación.

### 2.2.1. Parametros

- $I$ , El conjunto de elementos posibles, de cardinalidad  $N$
- $P_i$ , El beneficio asociado al elemento  $i$
- $LC_i$ , El coste nominal del elemento  $i$
- $UC_i$ , El coste máximo del elemento  $i$
- $W$ , El presupuesto o coste máximo asociado al problema.

- $S$ , El conjunto de sinergias polinomiales.
- $PS_A$ , El beneficio de la sinergia  $A$
- $\Gamma$ , El número máximo de elementos que pueden variar de su costo base.

### 2.2.2. Variables de decision:

$$x_i = \begin{cases} 1 & \text{si } i \in X \\ 0 & \text{si } i \notin X \end{cases} \quad (1)$$

$$Z_A = \begin{cases} 1 & \text{si } i \in A, \forall i \in X \\ 0 & \text{si } \exists i \in A : i \notin X \end{cases} \quad (2)$$

$$\rho, \pi_i \in \mathbb{R}^+ \quad (3)$$

### 2.2.3. Función objetivo

$$\text{Maximizar } \sum_i^I P_i x_i + \sum_{A \in S} PS_A \cdot Z_A - \sum_i^I LC_i x_i - \left( \Gamma \rho + \sum_i^I \pi_i \right) \quad (4)$$

### 2.2.4. Restricciones

$$\sum_i^I LC_i x_i + \Gamma \rho + \sum_i^I \pi_i \leq W \quad (5)$$

$$\rho + \pi_i \geq (UC_i - LC_i) \cdot x_i \quad \forall i \in I \quad (6)$$

$$\sum_{i \in A} x_i \leq |A| - 1 + Z_A \quad \forall A \in S : PS_A < 0 \quad (7)$$

$$Z_A \leq x_i \quad \forall i \in A \in S : PS_A > 0 \quad (8)$$



$$x_i \in 0, 1, \forall i \in I, \quad Z_A \in \{0, 1\} \forall A \in S \quad (9)$$

$$\rho \geq 0, \quad \pi_i \geq 0 \forall i \in I \quad (10)$$

La función objetivo en la ecuación 4 representa el beneficio de los beneficios y sinergias de los elementos restando el coste máximo posible que estos puedan tener.

La restricción de presupuesto se define con las ecuaciones 5 y 6, en principio  $X$  debe ser robusto a las variaciones en los costes por lo que se usa un acercamiento que asume el peor de los casos, es decir el cálculo del coste será siempre maximizado según las posibles variaciones. En este contexto para linealizar el problema Baldo et al. (2023) define las variables de decisión auxiliares 3, que transforman el coste máximo a un problema de minimización.

Para fijar la variable auxiliar  $Z_A$  se especifican las restricciones 7 y 8, que favorece las sinergias con beneficio positivo y desfavorece las con beneficio negativo.

## 2.3. Revisión de literatura

El trabajo de Baldo et al. (2023) ha introducido por primera vez una metodología para resolverlo, proponiendo un algoritmo genético y un algoritmo basado en machine learning. Este último utiliza un clasificador random forest que predice la probabilidad de cada elemento de estar presente en la solución óptima. Esta predicción se usa para fijar variables de decisión, reduciendo así el tiempo de ejecución de un solver exacto.

Li et al. (2021) ha estudiado el uso de redes neuronales para obtener predicciones de KPs con funciones objetivo no lineales, obteniendo buenos resultados con una estructura basada en teoría de juegos, junto al uso de redes neuronales adversarias.

Rezoug et al. (2022) aborda el KP usando distintas técnicas para evaluar las características de los elementos, entre ellos redes neuronales, regresión de procesos gaussianos, random forest y support vector regression. Así, resuelve el problema original, usando solo un subconjunto de los elementos. Luego, mediante el descenso del gradiente y el uso de características de los elementos, decide cuáles de los elementos excluidos agregar a la solución inicial obtenida. En sus resultados muestra que el modelo machine learning entrega soluciones similares a los otros clasificadores, con menores tiempos de ejecución.

Afshar et al. (2020) propuso un algoritmo para generar soluciones para el KP usando un modelo de Deep Reinforcement Learning que selecciona los elementos de forma greedy. El algoritmo propuesto construye las soluciones en base a las decisiones del modelo y genera soluciones con una razón de similitud con el óptimo del 99.9%. Además, usa una arquitectura de A2C con un paso de cuantización de características, que reduce la complejidad espacial del problema en la red, resultando en modelos que usan menos memoria y se ejecutan más rápido.

Si bien estos trabajos no se relacionan directamente con la variante del problema propuesto, sí evidencian que las técnicas de Machine learning pueden usarse de forma efectiva para caracterizar y construir soluciones para el PRKP.

## Capítulo 3

# Metodología

En este capítulo se expondrá el algoritmo propuesto para resolver el PKRP, así como las definiciones del modelo machine learning que actúa como clasificador, las características de los ítems que se usarán y el proceso de reducción de instancias.

### 3.1. Algoritmo propuesto

Se propone un algoritmo iterativo con el objetivo de reducir la complejidad del problema de forma gradual, a costa de un pequeño nivel de confianza sobre la solución final obtenida. El algoritmo usa una red neuronal como clasificador. Que con base en ciertas características de cada elemento, decide la probabilidad de estar o no en la solución final. Estas predicciones se usan para asumir ceros en la solución y reducir la instancia original a una más pequeña mediante una transformación. Este proceso puede ejecutarse un número arbitrario de veces, mientras se mantenga la precisión del modelo, al predecir instancias con cada vez menos elementos.

**Algoritmo 1** Algoritmo general

---

```

1:  $\tau$                                 ▷ Umbral de confianza del clasificador
2:  $I$                                 ▷ Conjunto de elementos inicial
3:  $S$                                 ▷ Conjunto de sinergias
4: loop
5:    $\mu \leftarrow \frac{|I|}{\log_{10} |I|}$     ▷ Maximo numero de elementos a bloquear por paso
6:    $F \leftarrow \text{Características}$     ▷ Cálculo de características de los elementos
7:    $\tilde{x} \leftarrow \text{Clasificador}(F)$   ▷ Generación de predicciones continuas
8:    $Z \subset I \leftarrow Z(\tilde{x}, \tau, \mu)$   ▷ Selección de elementos para eliminar
9:   if  $|Z| == 0$  or  $\min(\tilde{x}) > \tau$  then
10:    break
11:  end if
12:   $I \leftarrow I - Z$                 ▷ Reducción de la instancia
13:   $S \leftarrow S - \{A \in S, \exists i \in A : i \in Z\}$   ▷ Actualización de sinergias
14: end loop
15:  $X \leftarrow \text{SolverExacto}(I)$     ▷ Resolución exacta del conjunto reducido

```

---

El algoritmo 1 inicia con la definición del umbral de confianza del clasificador  $\tau$  en la línea 1. Se establece el conjunto inicial de elementos  $I$  en la línea 2, y el conjunto de sinergias  $S$  en la línea 3.

El bucle principal comienza calculando las características de los elementos en la línea 6. Posteriormente, el clasificador genera predicciones  $\tilde{x}$  en la línea 7 que se encuentran en el rango  $[0, 1]$ . A continuación, se selecciona un conjunto de elementos  $Z$  para eliminar en la línea 8 en función de las predicciones y el umbral  $\tau$ .

El tamaño maximo de  $Z$  se define por  $\mu$  en funcion del numero de elementos que existe en la instancia reducida, como se muestra en la linea 5.

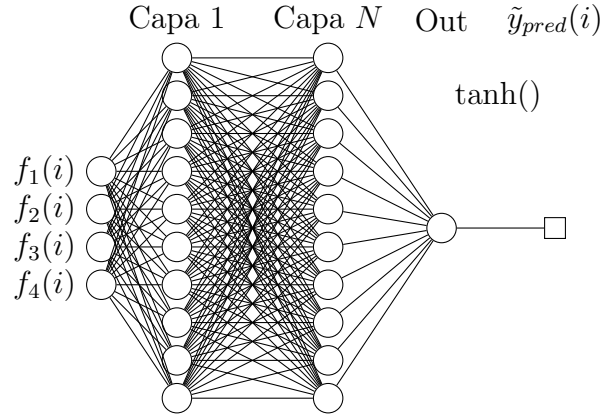
El bucle continúa hasta que el conjunto  $Z$  está vacío o el valor mínimo de las predicciones  $\tilde{x}$  supera el umbral  $\tau$ , como se verifica en la línea 9. Durante cada iteración, se eliminan los elementos de  $Z$  de  $I$  y se descartan todas las sinergias que incluyan algun elemento de  $Z$ .

Finalmente, el algoritmo resuelve exactamente el conjunto reducido  $I$  utilizando el *solver* exacto en la línea 15, y el resultado se almacena en  $X$ .

## 3.2. Clasificador

El clasificador utilizado es una red neuronal de cinco capas. Para la entrada usa features calculadas para un elemento de la instancia, así, se genera una predicción sobre si el

elemento tendrá un valor de cero o uno en la solución final.



**Figura 1:** Estructura general de la red

$$\tilde{x}(i) = \frac{1}{2} (\tilde{y}(i) + 1) \quad (11)$$

Como se muestra en la Figura la entrada de la red es un vector que contiene todas las features del elemento  $i$ :

$$f(i) = (f_1(i), f_2(i), \dots, f_\phi(i))$$

donde  $\phi$  es del número de features.

Através de la función de activación tangente hiperbólica, la red genera una salida  $\tilde{y}$  que se encuentra en el rango  $[-1,1]$ , la que es posible reconstruir usando la transformación de la ecuación 11.

En base a esto es útil definir la matriz  $F_{\phi \times N}$  como en la ecuación 12:

$$F_{\phi \times N}(I) = \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N-1) \\ f(N) \end{pmatrix} \quad (12)$$

Así, la red es una función definida como:

**Predicción para ítem:**

$$Net(f) = \tilde{x}_i : f \in [0, 1]^\phi \rightarrow \tilde{x}_i \in [0, 1]$$

**Predicción para todos los ítems de una instancia:**

$$Net(F) = \tilde{x} : F \in [0, 1]^{\phi \times N} \rightarrow \tilde{x} \in [0, 1]^N$$

Las features que se utilizan, codifican los parámetros  $P_i$ ,  $LC_i, UC_i$ ,  $\Gamma$ , además de,  $ContSol_i, Noise$ . Las definiciones se encuentran en el anexo [B.1](#).

Entre estas:

- $ContSol_i$  se refiere a la solución de la relajación continua del PRKP en que la variable de decisión  $x_i$  es continua.
- Noise es un generador de números aleatorios, que se usa para definir una entrada vacía en la red neuronal, esto da lugar para reemplazar esta feature por otra asociada al estado del algoritmo general.

### 3.2.1. Entrenamiento

Se realizan dos etapas de entrenamiento sobre un conjunto de instancias. La primera etapa consiste en entrenar la red para predecir la solución óptima de cada instancia, y así crear un clasificador general. En la segunda etapa, la red se afina con instancias generadas desde el algoritmo iterativo de reducción. Además, las soluciones con las que se entrenará la red son generadas por un solver exacto que calcula óptimo para cada instancia.

**Algoritmo 2** Primer entrenamiento

---

```

1: TrainingSet                                ▷ Conjunto de instancias de entrenamiento
2: Net                                          ▷ Red neuronal definida anteriormente
3: Optimizer                                  ▷ Optimizador para la red
4: Solver                                      ▷ Solver exacto para el problema
5: Loss                                        ▷ Funcion de perdida
6: for Instance  $\in$  TrainingSet do
7:    $I \leftarrow Instance_I$ 
8:    $x \leftarrow Solver(I)$                 ▷ Vector booleano  $x$  con la solución óptima de la instancia
9:   for  $i \in I$  do
10:     $\tilde{x}_i \leftarrow Net(f(i))$           ▷ Predicción de la red de  $x_i$  desde las features
11:     $loss \leftarrow Loss(x_i, \tilde{x}_i)$ 
12:     $Net \leftarrow Optimizer(Net, loss)$     ▷ Un paso de optimización de la red
13:   end for
14: end for
15:  $Net$                                      ▷ Red entrenada

```

---

El algoritmo 2 de entrenamiento comienza designando un conjunto de instancias en la línea 1 cada una de las estas define el conjunto de parámetros del modelo de programación lineal. Se define la red que se usará en la línea 2, inicializándose con pesos aleatorios. En la línea 5 se declara la funcion de perdida, que corresponde a la diferencia lineal entre el valor predicho y el valor real.

En la línea 3 se declara el optimizador encargado de reajustar los pesos de la red en cada paso de entrenamiento, se usa el algoritmo Adam introducido por Kingma and Ba (2017), que funciona como alternativa al descenso de gradiente estocástico. Finalmente se declara el solver exacto a usar, Gurobi 10.0.3.

A continuación, se inicia un bucle sobre cada instancia del conjunto de entrenamiento en la línea 6. Para cada instancia, se obtiene el conjunto de elementos asi como sus otros parametros, definiendo  $I$  de la instancia, en la línea 7. Se obtiene la solucion optima  $x$  para la instancia usando el solver en la línea 8.

Posteriormente, se inicia un bucle interno sobre los elementos de la instancia en la línea 9. Dentro de este bucle, la red neuronal realiza predicciones  $\tilde{x}_i$  sobre las características de la instancia utilizando la función  $f(i)$  en la línea 10. Se calcula la perdida entre la predicción y la solución óptima en la línea 11, y la red se actualiza mediante un paso de optimización utilizando Adam en la línea 12.

Este proceso de predicción, cálculo de pérdida y optimización se repite para todos los índices en el bucle interno. Una vez completado, se regresa al bucle externo para la siguiente

instancia del conjunto de entrenamiento.

Después de que se han procesado todas las instancias del conjunto de entrenamiento, la red neuronal  $Net$  se considera entrenada y se devuelve como resultado del algoritmo en la línea 15.

---

**Algoritmo 3** Segundo entrenamiento

---

```

1:  $\tau$                                 ▷ Umbral de confianza del clasificador
2: TrainingSet                        ▷ Conjunto de instancias de entrenamiento
3: Net                                ▷ Red neuronal pre-entrenada
4: Optimizer                          ▷ Optimizador para la red
5: Loss                               ▷ Función de perdida
6: for Instance  $\in$  TrainingSet do
7:    $I \leftarrow Instance_I$ 
8:   loop                                ▷ Bucle de reducción
9:      $\mu \leftarrow \frac{|I|}{\log_{10} |I|}$       ▷ Maximo numero de elementos a bloquear por ciclo
10:     $\tilde{x} \leftarrow Net(F(I))$           ▷ Vector de predicción de la red
11:     $Z \subset I \leftarrow Z(\tilde{x}, \tau, \mu)$ 
12:    if  $|Z| == 0$  or  $\min(\tilde{x}) > \tau$  then
13:      break
14:    end if
15:     $I \leftarrow I - Z$                                 ▷ Reducción de la instancia
16:     $S \leftarrow S - \{A \in S, \exists i \in A : i \in Z\}$     ▷ Actualización de sinergias
17:     $x \leftarrow Solver(I)$                             ▷ Solución óptima de la instancia reducida
18:     $\tilde{x} \leftarrow Net(F(I))$                           ▷ Prediccion para la instancia reducida
19:    for  $i \in I$  do
20:       $loss \leftarrow Loss(x_i, \tilde{x}_i)$                 ▷ Cálculo de la perdida
21:       $Net \leftarrow Optimizer(Net, loss)$               ▷ Actualización de los pesos de la red
22:    end for
23:  end loop
24: end for
25: Net                                ▷ Red entrenada

```

---

El segundo proceso de entrenamiento descrito en el algoritmo 3 sigue la estrategia del algoritmo general 1. Inicialmente se declaran los parámetros  $\tau$  y  $\mu$ , así como el conjunto de instancias de entrenamiento, la red neuronal pre-entrenada, el optimizador y la función de perdida, en las líneas 1 a 5. Se ejecuta el bucle de reducción por cada instancia, usando red la predicción en la línea 10. Al igual que en el algoritmo general 1, se usa esta predicción, para generar una instancia reducida, este proceso ocurre en las líneas 11 a 16. Luego se calcula la solución óptima para la instancia reducida, en la línea 17, así como la predicción de la red en la línea 18. Finalmente, se realiza un paso de optimización por cada elemento en la instancia reducida donde se calcula la perdida entre la predicción y el valor óptimo



en la línea 20 y se ajustan los pesos de la red en la línea 21. Después de ejecutar este proceso por todas las instancias del conjunto de entrenamiento, y actualizar los pesos usando todos los pasos de reducción de las instancias, se obtiene la red entrenada final.

Un cambio que mejora la precisión de la red resolver instancias reducidas, es reemplazar la feature 19 que correspondía un valor aleatorio, por una metrica que representa cuanto se ha reducido la instancia desde su tamaño original. Por esto en las ocasiones que se calcula  $F(I)$  para el segundo entrenamiento,  $f_6$  es reemplazada por la feature de la ecuacion 13, donde la instancia actual se refiere  $|I|$  de cada ciclo del bucle de reduccion.

$$f_7(i) = \frac{\text{N instancia Original}}{\text{N instancia actual}} \quad (13)$$

### 3.3. Reducción de instancias

De forma comprensiva, ya no es necesario considerar los elementos de  $Z$  en la solución, pero más importante, todas las sinergias polinomiales que poseían elementos en  $Z$ , ya no son alcanzables por ninguna solución, por lo que es seguro eliminarlas de  $S$ , notar también que  $\Gamma$  no es modificado.

Al aplicar esta transformación de forma iterativa es posible asumir ciertas propiedades:

- En cada paso, la solución óptima de cada iteración de la instancia estará compuesta por una proporción mayor de unos que de ceros. La proporción se le llamará, la densidad  $D$  de la instancia .
- Como  $\Gamma$  no cambia al eliminar ceros, es posible que se de la situación en que  $\Gamma > N$ , en cuyo caso el problema pasa de ser un PRKP a PKP, puesto que para asegurar la robustez, solo es necesario considerar los costes altos de la solución.
- Existe un punto en el que ya no quedan ceros que eliminar, el cual sucede cuando 5 se cumple para un  $x_i$  compuesto solo de unos, lo que es posible calcular a medida que se ejecuta la iteración.

Se define  $Z$  con base en la predicción del clasificador, si  $\tilde{x} = \text{Clasificador}(I)$ , la predicción continua del clasificador puede ser interpretada como un vector de probabilidades, donde valores cercanos a cero son interpretados como que el clasificador predice para ese extremo y equivalente para el las predicciones cercanas a uno.

Se definen entonces dos parametros para generar  $Z$ ,  $\tau$  y  $\mu$ :

- $\tau$  Corresponde al *threshold*, representa la confianza mínima necesaria, para decidir incluir un ítem en  $Z$
- $\mu$  Es el número máximo de elementos que incluir en  $Z$ .

A base de estos parámetros, el algoritmo 4 construye  $Z$

---

**Algoritmo 4**  $Z(\tilde{x}, \tau, \mu)$ 


---

```

 $A \leftarrow \{i \in I : x_i \leq \tau\}$ 
 $A \leftarrow \text{Sort}(A)$  ▷ Ordenar de forma ascendente en base a los valores de  $x_i$ 
 $Z \leftarrow \{\}$ 
 $count \leftarrow 0$ 
for all  $i \in A$  do ▷ Agregar los  $\mu$  ítems con menor  $x_i$ 
     $Z \leftarrow i$ 
     $count \leftarrow count + 1$ 
    if  $count == \mu$  then
        break
    end if
end for
 $Z$ 

```

---

El threshold  $\tau$  se establece por el usuario para definir la precision y velocidad de ejecucion del algoritmo y pertenece al rango de valores  $[0, 0.5]$ .

El número máximo de elementos que bloquear en cada paso  $Z$  se ha definido como  $Z = \max\{\frac{N}{\log_{10} N}, 100\}$  para cada instancia que se genera por reducción.

## Capítulo 4

# Resultados Experimentales

En este capítulo se presentará la comparación cuantitativa de resultados, se recontextualizarán los métodos de Baldo et al. (2023) con base en las diferencias de hardware y se mostrará la evaluación del algoritmo propuesto.

### 4.1. Instancias

Las instancias reales que se resuelven provienen de un generador de instancias implementado por Baldo et al. (2023) cuyas sinergias tienen en su mayoría beneficios de cero. Las instancias con  $I$  mayor a 1000, tienen una generación exponencial de sinergias. Para  $300 \leq I \leq 1000$ , se usa una generación cuadrática de sinergias. Para instancias con menos de 300 elementos, se generan sinergias de forma lineal.

Para comparar el rendimiento del algoritmo propuesto con los solvers existentes propuestos por Baldo et al. (2023), se usará el mismo conjunto de instancias.

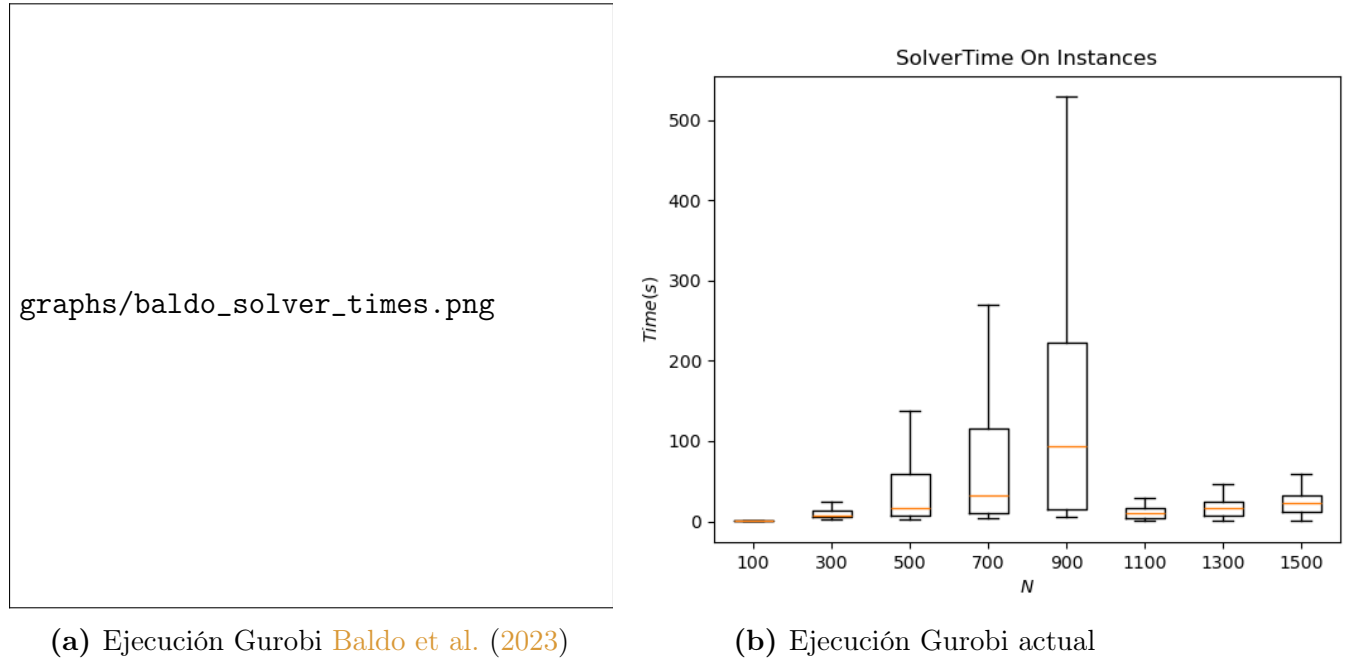
### 4.2. Comparaciones

Se calcularon los resultados para todas las instancias de Baldo et al. (2023), usando los métodos de BaldoGA y BaldoML, además de el algoritmo propuesto.

## Resultados generales

### Comparacion de solver exacto con trabajo anterior

En 2 se muestran los tiempos para resolver instancias usando el solver exacto, en contraste con Baldo et al. (2023), que solo resolvió instancias de hasta 650 elementos con tiempos hasta 7000s, en este caso los tiempos están acotados en cerca de 550 segundos.



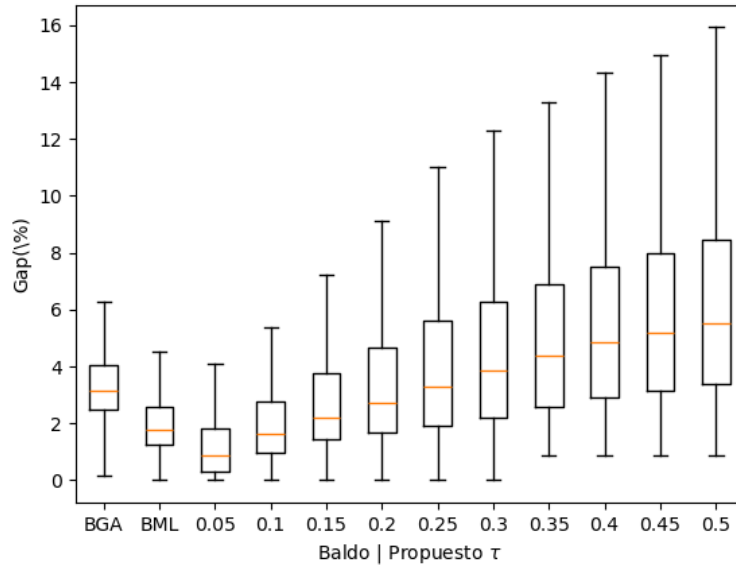
**Figura 2:** Tiempos de ejecución solver exacto

### Rendimiento normalizado

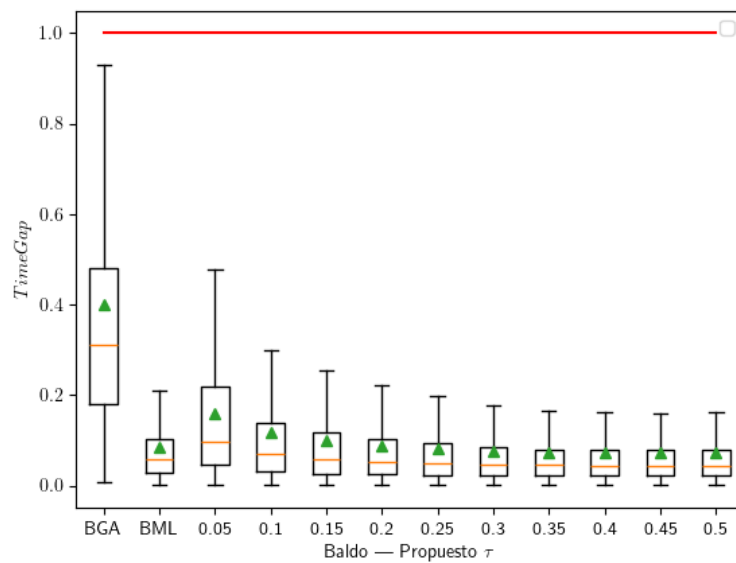
Se presenta en las figuras 3 y 4 la comparación general de rendimiento de los métodos. En primer lugar BaldoGA tienen el comportamiento esperado desde los resultados de Baldo et al. (2023), con un *gap* de alrededor de un (Insertar valor aquí :D). Por otro lado BaldoML tiene un *gap* promedio inferior al (Insertar valor aquí :D) esto es debido a diferencias en la inicialización del modelo randomforest al momento del entrenamiento, pero en mayor medida, por la diferencia de hardware, existe un subconjunto de instancias en las que el tiempo de ejecución del modelo, sumado al solver, superan el tiempo de ejecución del solver exacto, sobretodo al momento de calcular las features para la ejecución sobre las sinergias polinomiales.

Respecto a los tiempos de ejecución, se puede notar que el algoritmo propuesto con un  $\tau = 0.1$  genera resultados similares a BaldoGA manteniendo un 90% de los tiempos de

ejecución, menores a la media de los de BaldoGA. Además, BaldoML, a pesar de obtener resultados con un gap muy cercano a cero, sus tiempos de ejecución tienden a ser elevados y en la figura 4 se muestra visualmente que cerca de un 20 % de los tiempos de ejecución es superior al tiempo del solver exacto.



**Figura 3:** Gap % de todos los metodos presentados



**Figura 4:** TimeGap de los métodos usados

Método			N							
			100	300	500	700	900	1100	1300	1500
BaldoGA	Gap( %)	$\mu$	1.74	3.07	3.04	3.42	3.74	3.4	4.1	3.9
		$\sigma$	1.15	0.905	0.866	0.844	1.09	1.07	6.89	1.18
	Time(s)	$\mu$	0.44	3.3	4.5	7.18	9.86	5.61	7.22	8.89
		$\sigma$	0.207	1.32	1.75	2.93	4.08	2.19	2.44	3.21
BaldoML	Gap( %)	$\mu$	0.15	0.0723	0.0751	0.105	0.037	1.03	2.01	2.02
		$\sigma$	0.339	0.151	0.202	0.218	0.0736	10.0	14.0	14.0
	Time(s)	$\mu$	0.475	3.74	7.9	12.9	21.1	18.6	30.8	38.2
		$\sigma$	0.195	3.25	5.86	11.3	15.9	60.6	83.0	82.2
$\tau = 0.25$	Gap( %)	$\mu$	7.47	3.94	3.42	3.37	3.7	3.43	3.52	3.33
		$\sigma$	3.81	1.97	2.0	1.9	2.29	1.97	1.93	1.7
	Time(s)	$\mu$	0.0472	0.335	0.65	1.1	1.74	1.2	1.6	2.01
		$\sigma$	0.0202	0.118	0.278	0.383	0.655	0.614	0.818	0.904

**Tabla 3:** Comparación general por número de elementos

La tabla 3 es un extracto de la tabla completa de resultados 4, que muestra BaldoML, BaldoGA y El algoritmo propuesto con un  $\tau = 0.25$ . Es posible notar, que el algoritmo es menos preciso al encontrar soluciones de instancias de 100 elementos, pero se estabiliza con un gap promedio de alrededor de un 3 % para instancias de mayor tamaño. Siendo comparable a BaldoGA con tiempos de ejecución una orden de magnitud más pequeños.

## Capítulo 5

## Conclusiones

# Bibliografia

- Afshar, R. R., Zhang, Y., Firat, M., and Kaymak, U. (2020). A State Aggregation Approach for Solving Knapsack Problem with Deep Reinforcement Learning. In *Proceedings of The 12th Asian Conference on Machine Learning*, pages 81–96. PMLR. ISSN: 2640-3498.
- Baldo, A., Boffa, M., Cascioli, L., Fadda, E., Lanza, C., and Ravera, A. (2023). The polynomial robust knapsack problem. *European Journal of Operational Research*, 305(3):1424–1434.
- Eilon, S. (1987). Application of the knapsack model for budgeting. *Omega*, 15(6):489–494.
- Forrester, R. J. and Waddell, L. A. (2022). Strengthening a linear reformulation of the 0-1 cubic knapsack problem via variable reordering. *Journal of Combinatorial Optimization*, 44(1):498–517.
- Gallo, G., Grigoriadis, M., and Tarjan, R. (1989). A Fast Parametric Maximum Flow Algorithm and Applications. *SIAM J. Comput.*, 18:30–55.
- Gallo, G., Hammer, P. L., and Simeone, B. (1980). Quadratic knapsack problems. In Padberg, M. W., editor, *Combinatorial Optimization*, Mathematical Programming Studies, pages 132–149. Springer, Berlin, Heidelberg.
- Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.
- Kofler, C., Greistorfer, P., Wang, H., and Kochenberger, G. (2014). A Penalty Function Approach to Max 3-SAT Problems. *Working Paper Series, Social and Economic Sciences*. Number: 2014-04 Publisher: Faculty of Social and Economic Sciences, Karl-Franzens-University Graz.
- Li, D., Liu, J., Lee, D., Seyedmazloom, A., Kaushik, G., Lee, K., and Park, N. (2021). A Novel Method to Solve Neural Knapsack Problems. In *Proceedings of the 38th International Conference on Machine Learning*, pages 6414–6424. PMLR. ISSN: 2640-3498.
- Martello, S. and Toth, P. (1990). *Knapsack problems: algorithms and computer implementations*. Wiley-Interscience series in discrete mathematics and optimization. J. Wiley & Sons, Chichester ; New York.
- Mohammadi, S., Gleich, D. F., Kolda, T. G., and Grama, A. (2017). Triangular Alignment (TAME): A Tensor-Based Approach for Higher-Order Network Alignment. *IEEE/ACM transactions on computational biology and bioinformatics*, 14(6):1446–1458.

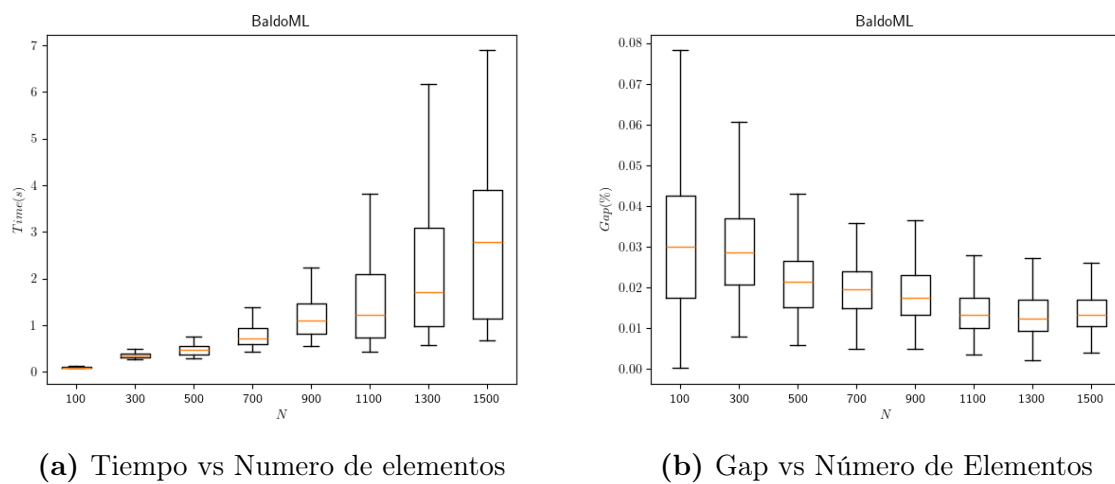


- Monaci, M., Pferschy, U., and Serafini, P. (2013). Exact solution of the robust knapsack problem. *Computers & Operations Research*, 40(11):2625–2631.
- Rezoug, A., Bader-el den, M., and Boughaci, D. (2022). Application of Supervised Machine Learning Methods on the Multidimensional Knapsack Problem. *Neural Processing Letters*, 54(2):871–890.
- Rhys, J. M. W. (1970). A Selection Problem of Shared Fixed Costs and Network Flows. *Management Science*, 17(3):200–207. Publisher: INFORMS.
- Witzgall, C. (1975). Mathematical methods of site selection for electronic message systems (EMS). Technical Report NBS IR 75-737, National Bureau of Standards, Gaithersburg, MD. Edition: 0.
- Zhao, K. (2008). Treatments of Chlamydia Trachomatis and Neisseria Gonorrhoeae. *Mathematics Theses*.

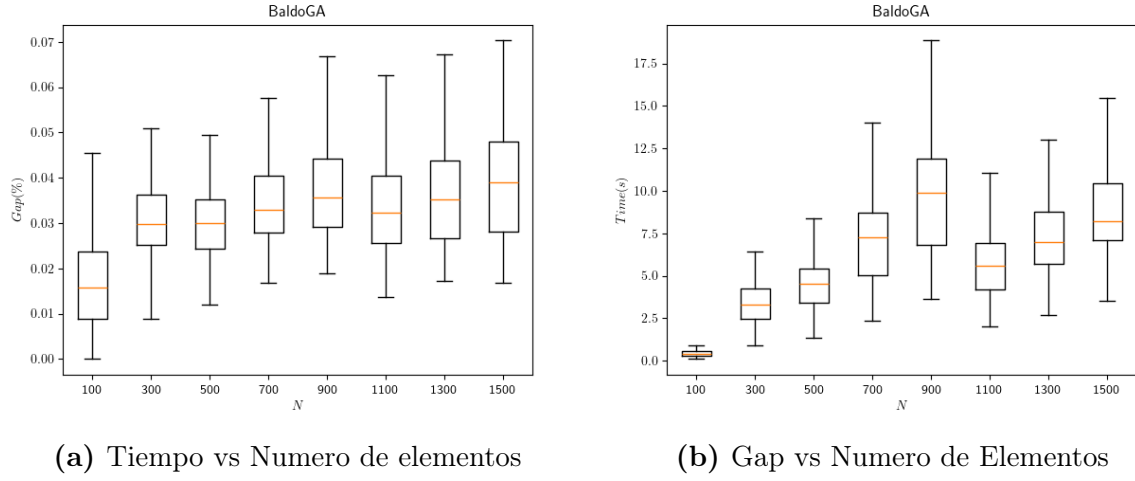
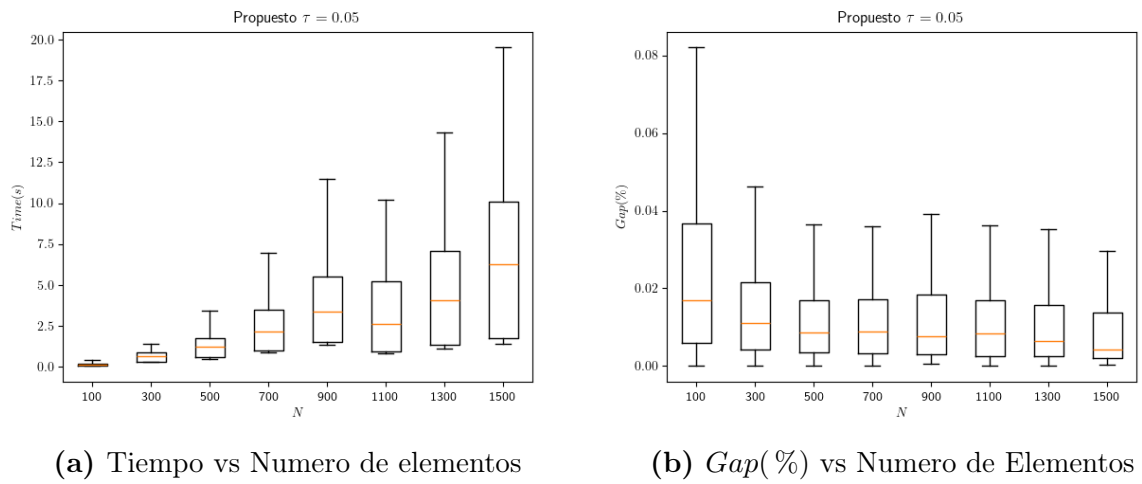
# Apéndice A

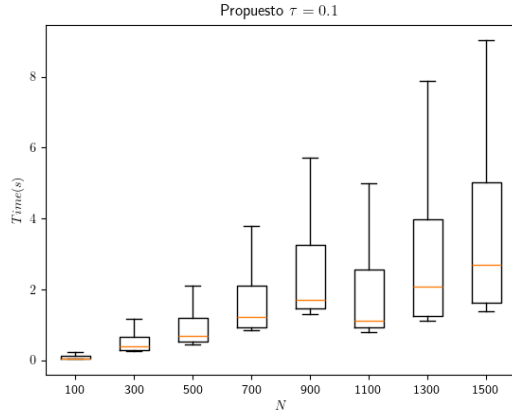
## Material

### A.1. Rendimiento de métodos en distintos tamaños de entrada

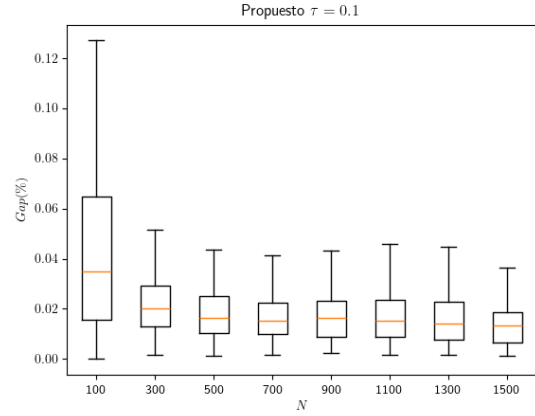
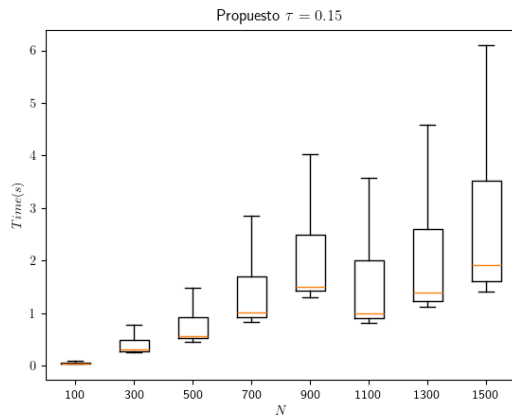


**Figura 5:** Rendimiento BaldoML

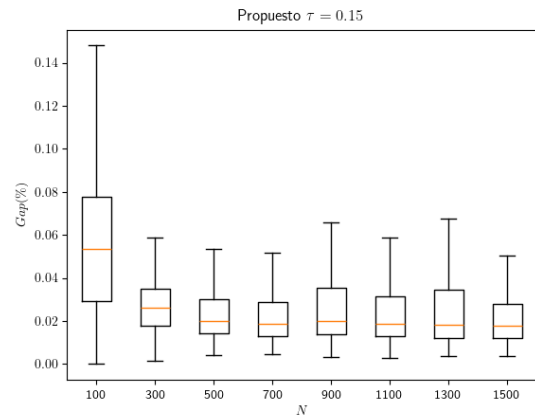
**Figura 6:** Rendimiento BaldoGA**Figura 7:** Rendimiento Algoritmo propuesto ( $\tau = 0.05$ )

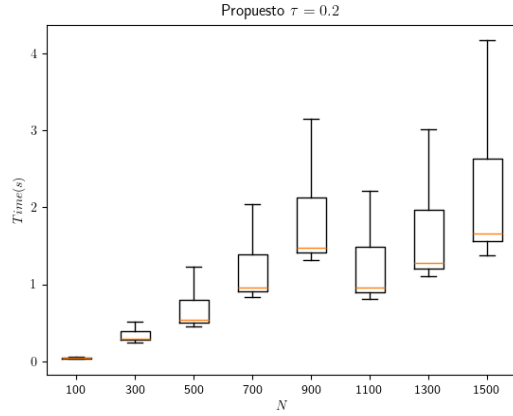


(a) Tiempo vs Numero de elementos

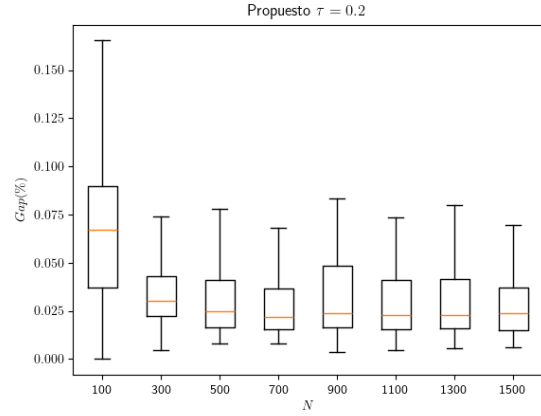
(b)  $Gap(\%)$  vs Numero de Elementos**Figura 8:** Rendimiento Algoritmo propuesto ( $\tau = 0.1$ )

(a) Tiempo vs Numero de elementos

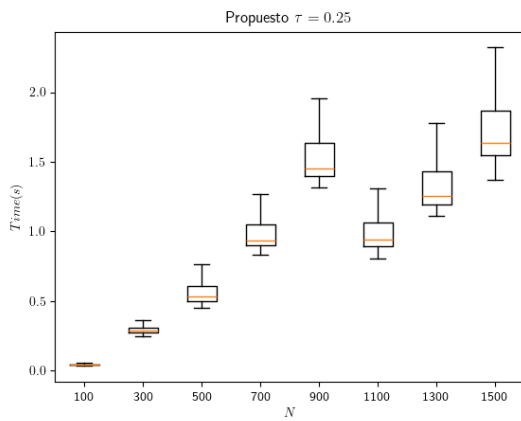
(b)  $Gap(\%)$  vs Numero de Elementos**Figura 9:** Rendimiento Algoritmo propuesto ( $\tau = 0.15$ )



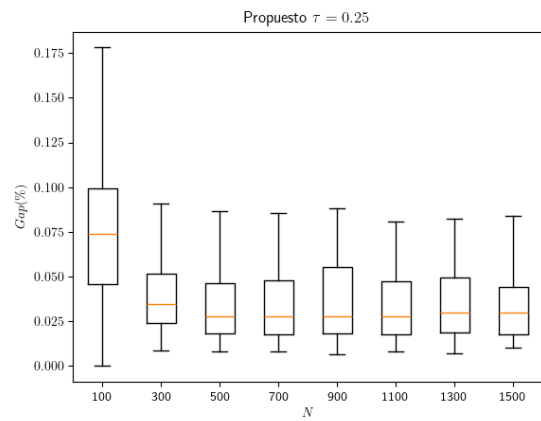
(a) Tiempo vs Numero de elementos



(b) Gap(%) vs Numero de Elementos

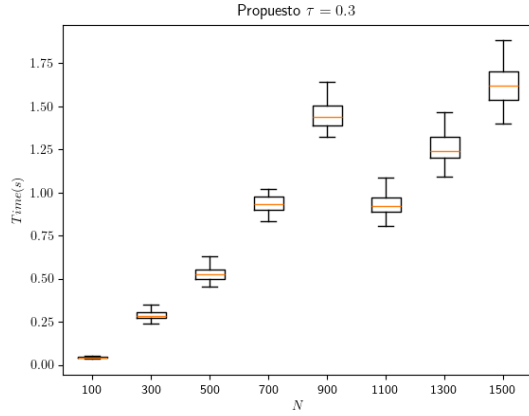
**Figura 10:** Rendimiento Algoritmo propuesto ( $\tau = 0.2$ )

(a) Tiempo vs Numero de elementos

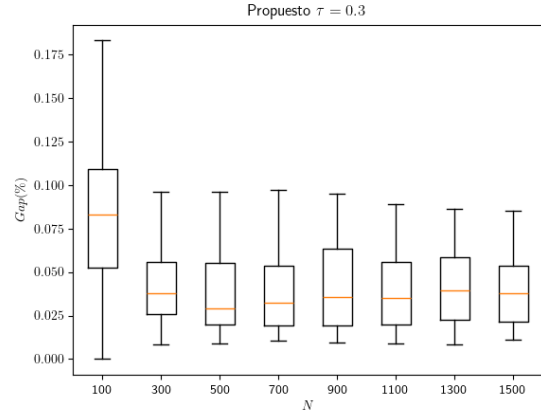
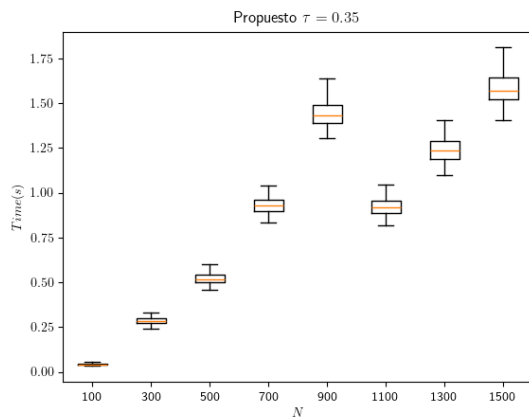


(b) Gap(%) vs Numero de Elementos

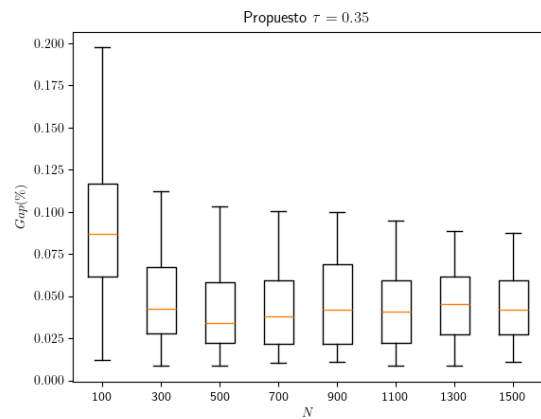
**Figura 11:** Rendimiento Algoritmo propuesto ( $\tau = 0.25$ )

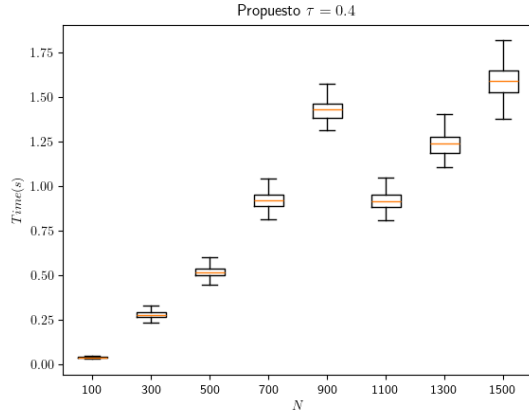


(a) Tiempo vs Numero de elementos

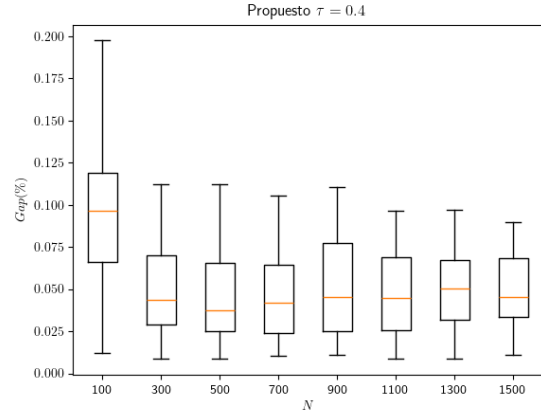
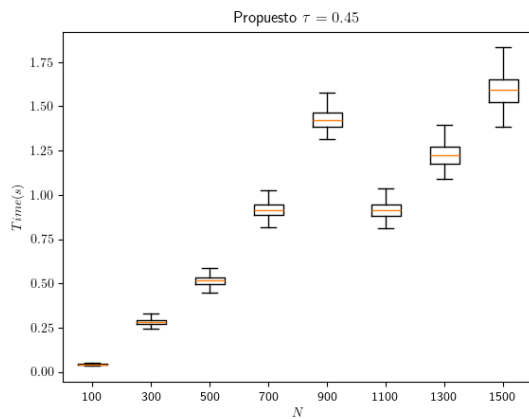
(b)  $Gap(\%)$  vs Numero de Elementos**Figura 12:** Rendimiento Algoritmo propuesto ( $\tau = 0.3$ )

(a) Tiempo vs Numero de elementos

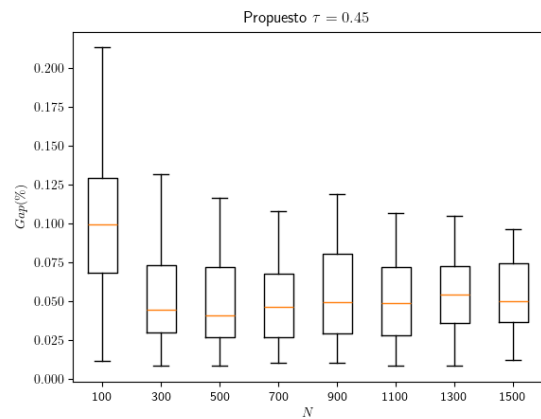
(b)  $Gap(\%)$  vs Numero de Elementos**Figura 13:** Rendimiento Algoritmo propuesto ( $\tau = 0.35$ )

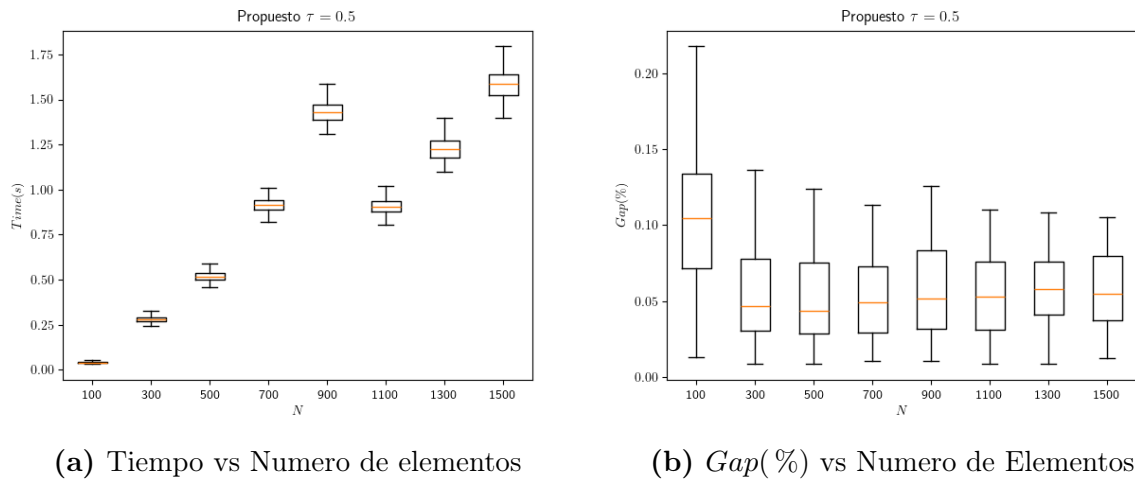


(a) Tiempo vs Numero de elementos

(b)  $Gap(\%)$  vs Numero de Elementos**Figura 14:** Rendimiento Algoritmo propuesto ( $\tau = 0.4$ )

(a) Tiempo vs Numero de elementos

(b)  $Gap(\%)$  vs Numero de Elementos**Figura 15:** Rendimiento Algoritmo propuesto ( $\tau = 0.45$ )



**Figura 16:** Rendimiento Algoritmo propuesto ( $\tau = 0.5$ )



Método			N							
			100	300	500	700	900	1100	1300	1500
BaldoGA	Gap( %)	$\mu$	1.74	3.07	3.04	3.42	3.74	3.4	4.1	3.9
		$\sigma$	1.15	0.905	0.866	0.844	1.09	1.07	6.89	1.18
	Time(s)	$\mu$	0.44	3.3	4.5	7.18	9.86	5.61	7.22	8.89
		$\sigma$	0.207	1.32	1.75	2.93	4.08	2.19	2.44	3.21
BaldoML	Gap( %)	$\mu$	0.15	0.0723	0.0751	0.105	0.037	1.03	2.01	2.02
		$\sigma$	0.339	0.151	0.202	0.218	0.0736	10.0	14.0	14.0
	Time(s)	$\mu$	0.475	3.74	7.9	12.9	21.1	18.6	30.8	38.2
		$\sigma$	0.195	3.25	5.86	11.3	15.9	60.6	83.0	82.2
$\tau_{0.05}$	Gap( %)	$\mu$	2.48	1.34	1.12	1.09	1.14	1.12	1.02	0.839
		$\sigma$	2.48	1.08	0.922	0.868	0.996	1.01	0.965	0.837
	Time(s)	$\mu$	0.138	0.665	1.32	2.46	4.08	3.43	4.84	6.68
		$\sigma$	0.104	0.391	0.874	1.74	2.98	2.8	3.83	4.69
$\tau_{0.1}$	Gap( %)	$\mu$	4.2	2.19	1.89	1.87	1.92	1.85	1.8	1.58
		$\sigma$	3.17	1.18	1.14	1.3	1.33	1.3	1.35	1.19
	Time(s)	$\mu$	0.0914	0.511	0.929	1.6	2.5	2.07	2.84	3.88
		$\sigma$	0.0735	0.281	0.562	0.848	1.52	1.7	2.12	3.11
$\tau_{0.15}$	Gap( %)	$\mu$	5.65	2.81	2.49	2.42	2.65	2.42	2.45	2.15
		$\sigma$	3.36	1.34	1.47	1.55	1.76	1.6	1.67	1.32
	Time(s)	$\mu$	0.0653	0.415	0.767	1.37	2.18	1.65	2.26	2.9
		$\sigma$	0.0491	0.207	0.392	0.66	1.25	1.22	1.6	1.97
$\tau_{0.2}$	Gap( %)	$\mu$	6.68	3.4	2.99	2.9	3.2	2.94	2.97	2.75
		$\sigma$	3.55	1.65	1.76	1.76	2.04	1.79	1.8	1.49
	Time(s)	$\mu$	0.0537	0.36	0.689	1.22	1.95	1.41	1.9	2.32
		$\sigma$	0.0313	0.148	0.3	0.527	1.03	0.933	1.28	1.25
$\tau_{0.25}$	Gap( %)	$\mu$	7.47	3.94	3.42	3.37	3.7	3.43	3.52	3.33
		$\sigma$	3.81	1.97	2.0	1.9	2.29	1.97	1.93	1.7
	Time(s)	$\mu$	0.0472	0.335	0.65	1.1	1.74	1.2	1.6	2.01
		$\sigma$	0.0202	0.118	0.278	0.383	0.655	0.614	0.818	0.904
$\tau_{0.3}$	Gap( %)	$\mu$	8.27	4.36	3.76	3.78	4.17	3.87	4.06	3.87
		$\sigma$	4.03	2.23	2.16	2.08	2.41	2.13	2.05	1.88
	Time(s)	$\mu$	0.044	0.317	0.611	1.04	1.58	1.06	1.41	1.79
		$\sigma$	0.0142	0.0979	0.232	0.299	0.407	0.426	0.538	0.574
$\tau_{0.35}$	Gap( %)	$\mu$	8.89	4.75	4.12	4.21	4.62	4.32	4.53	4.4
		$\sigma$	4.17	2.49	2.32	2.25	2.6	2.28	2.15	2.1
	Time(s)	$\mu$	0.0428	0.302	0.575	0.984	1.51	0.982	1.3	1.61
		$\sigma$	0.0112	0.0699	0.177	0.199	0.286	0.251	0.272	0.211
$\tau_{0.4}$	Gap( %)	$\mu$	9.5	5.02	4.45	4.57	5.04	4.71	4.96	4.88
		$\sigma$	4.26	2.67	2.49	2.4	2.75	2.43	2.27	2.27
	Time(s)	$\mu$	0.0413	0.295	0.55	0.943	1.45	0.938	1.24	1.59
		$\sigma$	4.950e-03	0.0571	0.128	0.123	0.161	0.163	0.077	0.0932
$\tau_{0.45}$	Gap( %)	$\mu$	10.0	5.27	4.77	4.94	5.42	5.05	5.32	5.27
		$\sigma$	4.35	2.89	2.67	2.56	2.88	2.58	2.38	2.43
	Time(s)	$\mu$	0.0416	0.292	0.535	0.921	1.43	0.918	1.23	1.59
		$\sigma$	4.670e-03	0.0485	0.0889	0.0716	0.0948	0.0513	0.0667	0.0893
$\tau_{0.5}$	Gap( %)	$\mu$	10.4	5.49	5.03	5.29	5.73	5.36	5.64	5.61
		$\sigma$	4.5	3.08	2.83	2.75	2.98	2.75	2.52	2.61
	Time(s)	$\mu$	0.0416	0.285	0.528	0.917	1.44	0.913	1.23	1.59
		$\sigma$	4.760e-03	0.0321	0.0676	0.0413	0.0745	0.0479	0.0702	0.0943

Tabla 4: Tabla completa de resultados

# Apéndice B

## Ecuaciones

### B.1. Features Clasificador

$$f_1(i) = \frac{P_i}{W} \quad (14)$$

$$f_4(i) = \frac{\Gamma}{N} \quad (17)$$

$$f_2(i) = \frac{LC_i}{W} \quad (15)$$

$$f_5(i) = \text{ContSol}_i \quad (18)$$

$$f_3(i) = \frac{UC_i}{W} \quad (16)$$

$$f_6(i) = \text{Random}(i) \quad (19)$$