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

Enero 2023

Profesor Guía: Carlos Contreras Bolton

Dedicatoria

AGRADECIMIENTOS

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer 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 consectetuer.

Resumen

Keywords – Knapsack Problem

Abstract

Keywords – Knapsack Problem

Índice general

Índice de Tablas

Índice de Figuras

Índice de Algoritmos

Capítulo 1

Introducción

Este capítulo presenta 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. Además, 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). Esta fue formulada originalmente por Eilon (1987) para resolver problemas de asignación de presupuesto con aplicaciones reales, el parámetro de costo del problema está asociado 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 costos inciertos y sinergias polinomiales para modelar problemas de selección de alternativas, que se perjudican o benefician entre sí y cuyos costos muestran comportamiento estocástico.

Debido a la dificultad para resolver el PRKP la literatura se ha limitado a explorar dos casos especiales del problema. El 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 el uso de una técnica de machine learning para resolver el PRKP. Una estrategia greedy que descarte elementos de las soluciones para acotar el espacio, siendo escalable y rápida de ejecutar en grandes tamaños de instancias, usando una red neuronal para la toma decisiones. Se compara con la literatura y se analizan los resultados obtenidos en instancias conocidas con soluciones conocidas y desconocidas de mayor tamaño.

1.2. Objetivos

Objetivo general

Implementar un algoritmo basado en machine learning para resolver el PRKP.

Objetivos específicos

- Revisar la literatura relacionada con problemas de la mochila similares y metodologías aplicables.
- Diseñar un algoritmo basado en machine learning para el PRKP.

- Implementar la estrategia propuesta basada en machine learning.
- Evaluar los resultados y comparar el rendimiento con las metodologías existentes en literatura.

1.3. Estructura del documento

El documento consta de seis capítulos, en los cuales se discute el problema y su formulación formal (2), la metodología propuesta para el PRKP (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 es definido formalmente como un conjunto de elementos $I = \{1, 2, \ldots, N\}$, 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, $C_i \in [C_i^L, C_i^U]$, donde C_i^L es el costo base, y C_i^U el costo máximo. Solo algunos elementos tienen comportamiento estocástico, aquellos que no, cumplen que $C_i = C_i^L$. El parámetro que define cuantos elementos de I pueden tener comportamiento estocástico es $\Gamma \leq |I|$, es decir $|\{i : C_i > C_i^L\}| \leq \Gamma$. Además, existe un presupuesto máximo W para los costes. Mientras, el conjunto de sinergias polinomiales se define como $A = \{\bar{I}_j \backslash \bar{I}_j \subseteq I : j \in 2^{|I|}\}$. Para cada sinergia $a \in A$ existe un beneficio asociado P_a. Finalmente, el objetivo del problema es encontrar $X \subseteq I$ 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, mantenga un coste total menor al presupuesto para cualquier variación estocástica en costes de los elementos.

Sea un ejemplo de instancia de PRKP con $I = \{1, 2, 3\}, W = 10$ y $\Gamma = 2$. La Tabla 1 describe las cotas inferior y superior de los costos de los elementos de la instancia, así como los beneficios de cada elemento individual. La Tabla 2 muestra los beneficios de cada sinergia. El número de sinergias es siempre igual a $2^|I| - |I|$, aunque en el caso de que hayan sinergias cuyo beneficio es cero, estas se omiten de la tabla. Una solución factible de esta instancia es $X = \{1, 3\}$ cuyo beneficio total es 6, mientras que la solución óptima de esta instancia, es $X = \{2, 3\}$ con un beneficio total de 9.

Tabla 1: Beneficios y costos de los ítems.

i	1	2	3
PS_i	3	6	10
C_i^L	2	3	4
C_i^U	4	4	5

Tabla 2: Beneficios por sinergias.

A	P_A
$\{1, 2\}$	-2
$\{1, 3\}$	2
$\{2, 3\}$	2
$\{1, 2, 3\}$	3

2.2. Modelo matemático

Baldo et al. (2023) describe una formulación del PRKP mediante un modelo matemático linealizado que es compatible con solvers de propósito general. El modelo de programación lineal entera mixta (MILP) usa una variable de decisión binaria $x_i : i \in I$ que toma el valor 1 si un elemento pertenece a la solución final. Las variables de decisión ρ y π_i representan la diferencia entre el coste base y el coste máximo de un elemento. Se ajustan de forma que si el coste del elemento i varía entonces la suma entre ρ y π_i es igual al coste máximo. Así la función objetivo se encarga de maximizar ρ y π_i para que considere el máximo coste posible para todos los elementos.

Tabla 3: Resumen de los conjuntos, parámetros y variables de decisión

Conjuntos	
I	El conjunto de elementos { 1,2,. . . ,N}.
A	El conjunto de sinergias $\{I_j \backslash I_j \subseteq I : j \in 2^{\mid}I\mid\}$.
Parámetros	
P_i	El beneficio asociado al elemento $i \in I$.
C_i^L	El coste base del elemento $i \in I$.
C_i^U	El coste máximo del elemento $i \in I$.
W	El presupuesto o coste total máximo asociado al problema.
B_a	El beneficio de la sinergia polinomial arbitraria $a \in A$.
Γ	El número máximo de elementos que pueden tener costo distinto al costo base.
Variables de decisión	
x_i	Toma valor 1 si el elemento $i \in I$ está presente en la solución óptima.
z_a	Toma valor 1 si $\forall a \in A, a \in X$, si no 0.
ρ, π_i	Cumplen que $\rho + \pi_i = C_i^U - C_i^L$ si el coste del elemento i varía, sino $\rho = C_i^U - C_i^L$ y $\pi_i = 0$.

$$\text{Maximizar} \quad \sum_{i \in I} P_i x_i + \sum_{a \in A} B_a z_a - \sum_{i \in I} LC_i x_i - \left(\Gamma\rho + \sum_{i \in I} \pi_i \right) \tag{1}$$

$$\sum_{i \in I} C_i^L x_i + \Gamma\rho + \sum_{i \in I} \pi_i \leq W \tag{2}$$

$$\rho + \pi_i \geq (C_i^U - C_i^L) x_i \qquad \forall i \in I \tag{3}$$

$$\sum_{i \in a} x_i \leq |a| - 1 + z_a \qquad \forall a \in A : B_a < 0 \tag{4}$$

$$z_a \leq x_i \qquad \forall i \in a, \forall a \in A : B_a > 0 \tag{5}$$

$$x_i \in \{0, 1\} \qquad \forall i \in I \tag{6}$$

$$z_a \in \{0, 1\} \qquad \forall a \in A \tag{7}$$

$$\rho \in \mathbb{R}^+ \quad \pi_i \in \mathbb{R}^+ \qquad \forall i \in I \tag{8}$$

La función objetivo (1) representa el beneficio total de elementos y sinergias de los elementos restando el coste máximo posible que estos puedan tener. $\sum_{i \in I} P_i x_i$ es la suma de todos los beneficios individuales de cada elemento. $\sum_{a \in A} B_a z_a$ corresponde a la suma de los beneficios de cada sinergia $\sum_{i \in I} C_i^L x_i$ es la suma de todos los costes mínimos y corresponde a la suma de la diferencia de los costes. $\left(\Gamma\rho + \sum_{i \in I} \pi_i\right)$. Las ecuaciones (2) y (3) representan la restricción de presupuesto. La solución debe ser robusta 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 es siempre maximizado según las posibles variaciones. Para ello Baldo et al. (2023) genera el problema dual de la maximización de los costes, construyendo la dualidad de Lagrange. Esto resulta en las restricciones (2) y (3) y las variables de decisión auxiliares ρ y $\pi_i (i \in I)$, de forma que el cálculo del coste se expresa como un problema de minimización. Las restricciones (4) y (5) aseguran que el valor de z estableciéndolo como uno si todos elementos de una sinergia están en la solución final y cero en caso contrario. La restricción (8) establece ρ y π_i números reales positivos, así el modelo no establece costes negativos en la función de maximización y en el cálculo del coste. A través de la restricción (3).

2.3. Revisión de literatura

El trabajo de Baldo et al. (2023) introduce el PRKP y 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. Como solo existe un trabajo para el PRKP, esta revisión se enfoca en metodologías de solución basadas en machine learning para variantes del KP. Li et al. (2021) utiliza redes neuronales para obtener predicciones de soluciones para instancias de KP que poseen funciones objetivo no lineales desconocidas. De esta forma solo se conocen parcialmente algunos valores de la función objetivo pero no su formulación. Los autores obtienen buenos resultados con una estructura basada en teoría de juegos, junto al uso de redes neuronales adversarias, considerando una red para modelar la función objetivo y otra con el método de descenso de gradiente adaptativo para resolver el problema.

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. Ellos resuelven el problema original usando solo un subconjunto de los elementos. Luego mediante el descenso del gradiente y el uso de características de los elementos deciden cuáles de los elementos excluidos debe agregar a la solución inicial obtenida. Los resultados muestran que el modelo machine learning entrega soluciones similares a los otros clasificadores y 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 dos redes neuronales con un paso de reducción de dimensión de características que reduce la complejidad espacial del problema en la red. Esto resulta en modelos que usan menos memoria y se ejecutan más rápido.

El uso de técnicas basadas en machine learning para abordar variantes del KP en la literatura, evidencian la posibilidad de diseñar estrategias efectivas que resuelvan el PRKP de forma eficiente.

Capítulo 3

Metodología

Este capítulo presenta el algoritmo propuesto para resolver el PKRP, así como las definiciones del modelo de machine learning que actúa como clasificador, las características de los elementos que se usan 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 del nivel de confianza sobre la solución final obtenida. El algoritmo usa una red neuronal como clasificador, que se basa en características de cada elemento para decidir la probabilidad de estar o no en la solución final. El clasificador es entrenado antes del algoritmo general para obtener predicciones adecuadas. Estas predicciones son usadas para que los elementos tomen valores de 0 % de probabilidad y así reducir la instancia original a una más pequeña. Este proceso es ejecutado un número arbitrario de veces, mientras se mantenga la precisión del modelo, al evaluar instancias con cada vez menos elementos. El enfoque define una estructura para la red, una serie de características para elemento, dos etapas de entrenamiento y un algoritmo de evaluación que usa el clasificador entrenado para evaluar y encontrar soluciones para las instancias. La construcción del algoritmo propuesto

3.2. Clasificador

El clasificador utilizado es una red neuronal de tres capas de 50 nodos cada una, con una salida, y una función de activación. Estructurada para tomar como entrada las

características de un elemento y obtener la predicción.

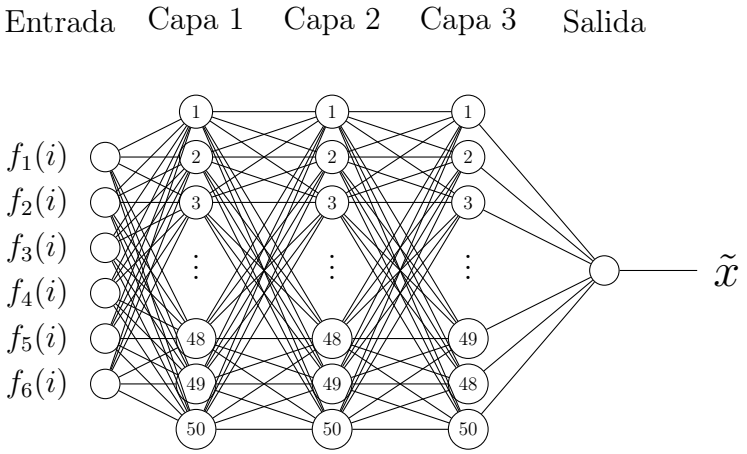

Figura 1: Estructura general de la red neuronal.

La Figura 1 muestra la entrada de la red neuronal como un vector que contiene todas las características del elemento $i \in I$ y genera un número en la capa de salida. La función de activación es tangente hiperbólica que se aplica a la capa de salida de la red para ajustar la predicción a al rango [-1,1]. Luego,la predicción para cada elemento está dada por la transformación la Ecuación (9).

$$\tilde{x}(i) = \frac{1}{2}\left(\tanh(Out(i)) + 1\right) \tag{9}$$

Así, la red toma como argumento el vector de características $\texttt{f}(i) = \big(f_1(i), f_2(i), f_3(i), f_4(i), f_5(i), f_6(i)\big)$, para retornar x_i, la predicción para el elemento i, entre cero y uno. Además, se puede extender como en la Ecuación (11) que toma como entrada una matriz $F(I)$ como definida en la Ecuación (12) que contiene las características de todos los elementos y obtiene como resultado el vector $\tilde{x}$ que contiene las probabilidades $\tilde{x_i}(\forall i \in I)$, que sirve como predicción de solución para la instancia.

$$Net(f) = \tilde{x_i} \quad : f \in [0,1]^6 \rightarrow \tilde{x}_i \in [0,1] \tag{10}$$

$$Net(F) = \tilde{x} \quad : F \in [0,1]^{6\times|I|} \rightarrow x \in [0,1]^{|I|} \tag{11}$$

$$F(I) \in M_{6\times|I|} = \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(|I|-1) \\ f(|I|) \end{pmatrix} \tag{12}$$

$$f_1(i) = \frac{P_i}{W} \tag{13}$$

$$f_2(i) = \frac{C_i^L}{W} \tag{14}$$

$$f_3(i) = \frac{C_i^U}{W} \tag{15}$$

$$f_4(i) = \frac{\Gamma}{|I|} \tag{16}$$

$$f_5(i) = \text{ContSol}_i \tag{17}$$

$$f_6(i) = \frac{|I|_{\text{Actual}}}{|I|_{\text{Original}}} \tag{18}$$

Entre estas:

- La característica en la ecuación (13), representa el costo de un elemento normalizado al presupuesto total disponible para en la instancia del problema.
- Las ecuaciones (14) y (15), son el costo base y el costo máximo respectivamente, normalizados usando el presupuesto.
- La característica de la ecuación (16), representa el porcentaje de los elementos que varían estocásticamente su coste. Es una característica agregada de todos los elementos de una instancia que es invariable del elemento.
- $ContSol_i$ se refiere a la solución de la relajación continua del PRKP en que la variable de decisión x_i en el modelo de programación lineal es continua.
- La característica de la ecuación (18) es un indicador de cuanto se ha reducido la instancia en cada iteración del Algoritmo 4, tiene un valor de 1 en la primera iteración y va decreciendo a medida que se eliminan elementos a considerar del conjunto I.

3.2.1. Entrenamiento

La primera etapa consiste en entrenar la red para predecir la solución óptima de cada instancia, y así crear un clasificador general. La segunta etapa afina la precisión usando

de los pasos intermedios del algoritmo de evaluación (4). Además, la red se entrena con soluciones que son generadas por un solver exacto que calcula el óptimo para cada instancia.

Algoritmo 1 Entrenamiento

Salida `Net`

1: `Net`
2: **for** Instance $\in$ TrainingSet **do**
3: $I \leftarrow Instance$
4: $x \leftarrow$ `Solver`(I)
5: **for** $i \in I$ **do**
6: $\tilde{x_i} \leftarrow$ `Net(f`$(i))$
7: $loss \leftarrow$ `Loss`$(x_i, \tilde{x_i})$
8: $Net \leftarrow$ `Optimizer(Net,`$loss$`)`
9: **end for**
10: **end for**

El Algoritmo 1 de entrenamiento comienza inicializando la red con pesos aleatorios en la línea 1. A continuación, se inicia un bucle itera sobre cada instancia del conjunto de entrenamiento en la línea 2. Para cada instancia, se obtiene el conjunto de elementos I de la instancia, en la línea 3. Se obtiene la solución óptima x para la instancia usando el solver en la línea 4. Posteriormente, se inicia un ciclo interno sobre los elementos de la instancia, donde la red neuronal realiza predicciones $\tilde{x_i}$ sobre las características de cada elemento utilizando la función $f(i)$ en la línea 6. Usando esta predicción se calcula la perdida entre la solución real x_i y la predicción $\tilde{x_i}$ del elemento. Los pesos de la red se actualizan usando el algoritmo de optimización Adam, introducido por Kingma and Ba (2017), que funciona como alternativa al descenso de gradiente estocástico, usando la pérdida anteriormente calculada. El 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. Habiendo procesado todas las instancias del conjunto de entrenamiento, la red neuronal Net se considera entrenada y el algoritmo termina.

Algoritmo 2 Afinación

Input: τ `Net`
Output: `Net`
1: **for** Instance ∈ TrainingSet **do**
2: $I \leftarrow$ Instance
3: **loop**
4: $\tilde{x} \leftarrow \texttt{Net}(F(I))$
5: $Z \leftarrow \texttt{Z}(I, \tilde{x}, \tau)$
6: **if** $|Z| = 0$ **or** mín$(\tilde{x}) > \tau$ **then**
7: **break**
8: **end if**
9: $I \leftarrow I \setminus Z$
10: $S \leftarrow \{A \in S, \forall i \in A : i \notin Z\}$
11: $x \leftarrow \texttt{Solver}(I)$
12: $\tilde{x} \leftarrow \texttt{Net}(F(I))$
13: **for** $i \in I$ **do**
14: $loss \leftarrow \texttt{Loss}(x_i, \tilde{x_i})$
15: $\texttt{Net} \leftarrow \texttt{Optimizer}(\texttt{Net}, loss)$
16: **end for**
17: **end loop**
18: **end for**

El Algoritmo 2 describe la etapa de afinación y sigue una estructura similar a la del Algoritmo 4. El ciclo principal itera sobre instancias del conjunto de entrenamiento y define el conjunto de elementos de la instancia en la linea 2. La instancia es sujeta al proceso de reducción del Algoritmo 4 entre las lineas 4 y 10. Al finalizar cada reducción se obtiene la solución óptima de la instancia reducida y la predicción de la red en las líneas 11 y 12. La solución óptima y la predicción de la red son usadas para calcular la perdida de cada elemento de la instancia reducida y actualizar los pesos de la red usando el algoritmo de optimización Adam. Este proceso afina la red para mejorar su precisión en instancias que han sido generadas por la reducción. Al finalizar la reducción y entrenamiento consecutivos con cada instancia del conjunto de entrenamiento, la red se encuentra entrenada para su uso en el Algoritmo 4 de evaluación.

3.3. Reducción de instancias

El proceso de reducción descrito en el Algoritmo 4 requiere definir el proceso de creación de Z con base en los parámetros. El Algoritmo 3 construye Z de forma que contenga los elementos ordenados con predicciones mas cercanas a cero que estén bajo el umbral τ,

con un tamaño máximo de μ. Comienza estableciendo el tamaño maximo de Z, μ, que se calcula como el numero de elementos de la instancia, divido sobre su magnitud. Si la instancia posee 10.000 elementos,como maximo Z tendrá 2.500. Luego se crea el conjunto B que posee todos los elementos cuya predicción $\tilde{x}_i$ es menor que τ en la línea 2. Luego en la línea 3 B es ordenado de manera ascendente, de forma que $\tilde{x}_{B_b} < \tilde{x}_{B_{b+1}} \forall b \in [1, |B|]$. Así, en un bucle que itera sobre los elementos ordenados de B, estos son agregados a Z en la linea 6. Este ciclo termina cuando B se queda sin elementos que agregar, o si la cardinalidad de Z supera el número máximo elementos definido por μ.

Algoritmo 3 $Z(I, \tilde{x}, \tau)$

Input: I,$\tilde{x}$,τ
Output: Z
1: $\mu \leftarrow \frac{|I|}{\log_{10}|I|}$
2: $B \leftarrow \{i \in I : \tilde{x}_i \leq \tau\}$
3: $B \leftarrow \texttt{Sort}(B)$
4: $Z \leftarrow \{\}$
5: **for all** $b \in B$ **do**
6: $Z \cup \{b\}$
7: **if** $|Z| = \mu$ **then**
8: **break**
9: **end if**
10: **end for**

3.4. Resolución de Instancias

Algoritmo 4 Algoritmo de evaluación

Input: τ
Output: x
1: **loop**
2: $F \leftarrow \texttt{F}(I)$
3: $\tilde{x} \leftarrow \texttt{Net}(F)$
4: $Z \leftarrow \texttt{Z}(I, \tilde{x}, \tau)$
5: **if** $|Z| = 0$ **or** $\min(\tilde{x}) > \tau$ **then**
6: **break**
7: **end if**
8: $I \leftarrow I \setminus Z$
9: $S \leftarrow \{A \in S, \forall i \in A : i \notin Z\}$
10: **end loop**
11: $x \leftarrow \texttt{Solver}(I)$

El Algoritmo 4 se encarga de usar una red pre-entrenada como clasificador y resolver la instancia. Comienza dentro de un ciclo, el primer paso en la línea 2 es generar características derivadas de los parámetros del modelo, que describan a cada elemento en particular. Estas características son usadas por la red en la línea 3 para generar la predicción de cada elemento. El conjunto de elementos de la instancia, la predicción de la red y el parámetro τ son usados en la línea 4 para generar Z, el conjunto de elementos a eliminar de la solución final. τ define el mínimo valor que debe tener la predicción de un elemento, para ser ignorado en la solución final. Todos los elementos de Z se quitan de la instancia, así como las respectivas sinergias polinomiales en las líneas 8 y 9. Este proceso se ejecuta hasta que Z se construye sin elementos, o hasta que ningún elemento consigue una probabilidad de ser cero menor a τ. Finalmente, en la linea 11 se usa un solver exacto para resolver la instancia que contiene un número reducido de elementos.

Capítulo 4

Resultados Experimentales

Este capítulo se presenta la comparación cuantitativa de los métodos de la literatura y el método propuesto.

4.1. Instancias

Las instancias provienen de un generador 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 ($|\mathrm{S}| \sim aI^{a|I|}$). Para $300 \leq I \leq 1000$, se usa una generación cuadrática de sinergias ($|\mathrm{S}| \sim aI^2$). Para instancias con menos de 300 elementos, se generan sinergias de forma lineal ($|\mathrm{S}| \sim a|I|$).

Para comparar el rendimiento del algoritmo propuesto con los métodos de Baldo et al. (2023) se usa el mismo conjunto de instancias de sus experimentos. Además, se generan 50 nuevas instancias con $|I| = 20000$ con el objetivo de comparar rendimiento para problemas de un orden de magnitud más grande entre BaldoML y el método propuesto.

4.2. Software y Hardware

La implementación del algoritmo se desarrolló usando Python 3.11.6, usando como librerías principales Pytorch 2.1, Numpy 1.26, y Gurobipy como binding para Gurobi 10.0.3. Los resultados fueron calculados en Linux 6.6.8 en un Intel i7-8550U de 4 núcleos, 8 hilos a 4.2 Ghz y 32 GB de RAM. La implementación se encuentra disponible en Github (https://github.com/elMixto/PRKP).

4.3. Comparaciones

Se calcularon los resultados para todas las instancias de Baldo et al. (2023), usando los métodos BaldoGA, BaldoML y el algoritmo propuesto usando $\tau \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$.

Resultados de entrenamiento

Para entrenar el clasificador se usaron 2500 instancias provenientes del generador de Baldo et al. (2023). Estas tienen tamaños que varían entre 100 y 1500 elementos de forma no uniforme y con Γ establecido de forma aleatoria entre cero y el numero elementos de cada instancia. El proceso de optimización se realizó 2 veces por cada instancia usando un learning rate de 10^{-5}.

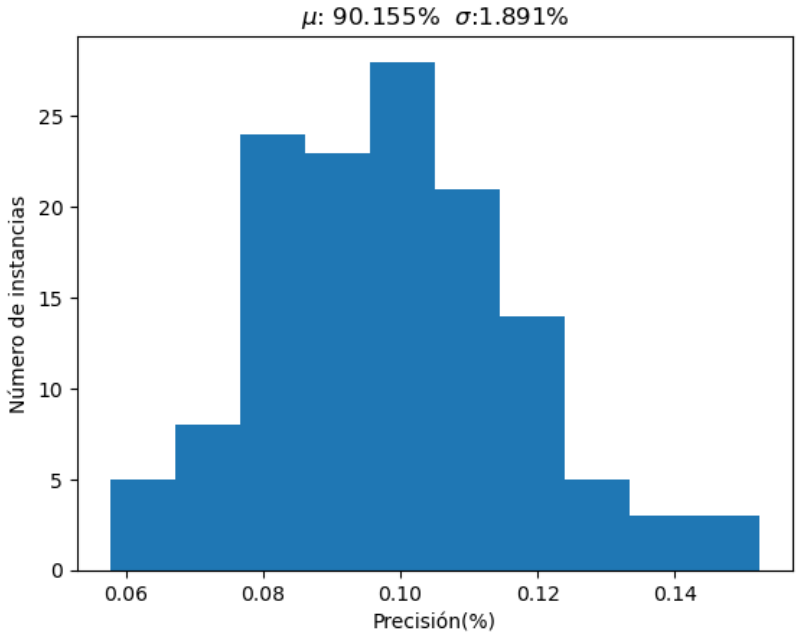

Figura 2: Distribución de la precisión del clasificador sobre instancias de evaluación

La Figura 2 muestra los resultados del entrenamiento. Se aplicó el modelo a un conjunto de evaluación de 135 instancias con tamaños entre 100 y 2000 elementos con Γ aleatorio, alcanzando una precisión de 90,155 % con una desviacion estandar de 1,891 %. Es decir, en promedio, el clasificador puede predecir correctamente si un elemento está o no en la solución final de una instancia, un 90 % del tiempo.

Resultados generales

La Figura 3 presenta la distribución del Gap(%) de cada método. BaldoGA tiene un gap promedio de $3,29\,\%$ y BaldoML tiene un gap promedio del $2{,}1\,\%$. Estos resultados son congruentes con el estudio de Baldo et al. (2023), mostrando comparabilidad con los resultados. El método propuesto muestra un aumento en la variación del Gap(%) conforme τ varía entre 0,05 y 0,25. Sin embargo, $\tau = 0.05$ muestra una distribucion con menor media y variacion del Gap(%) que a BaldoML y BaldoGA en todas las instancias. Respecto a los tiempos de ejecución. Por otro lado, la Figura 4 describe el TimeGap(%) de los métodos usados, en los que el tiempo de ejecución está normalizado usando el tiempo de solucion del solver exacto. BaldoGA posee el peor rendimiento de los metodos, consiguiendo gaps mas altos y usando en mucho mas tiempo que el resto de métodos. El método propuesto con $\tau = 0.05$ no supera el $40\,\%$ del tiempo que tardó el solver.

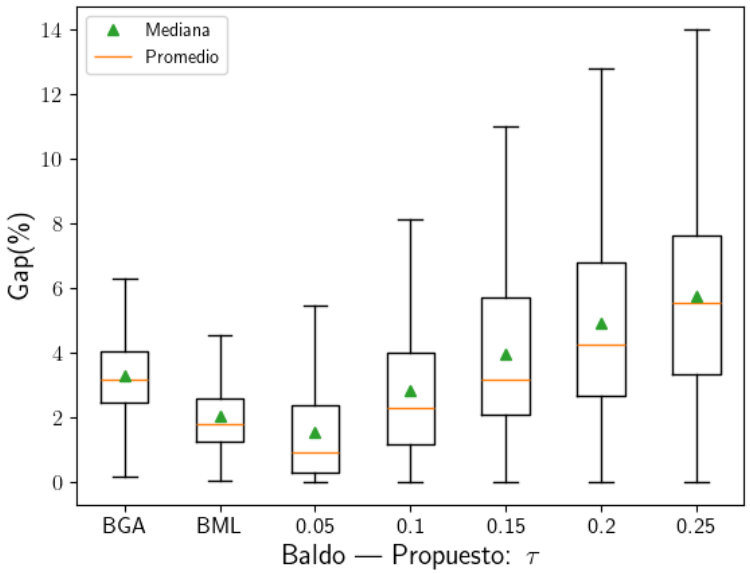

Figura 3: Distribución del Gap(%) de cada método presentado

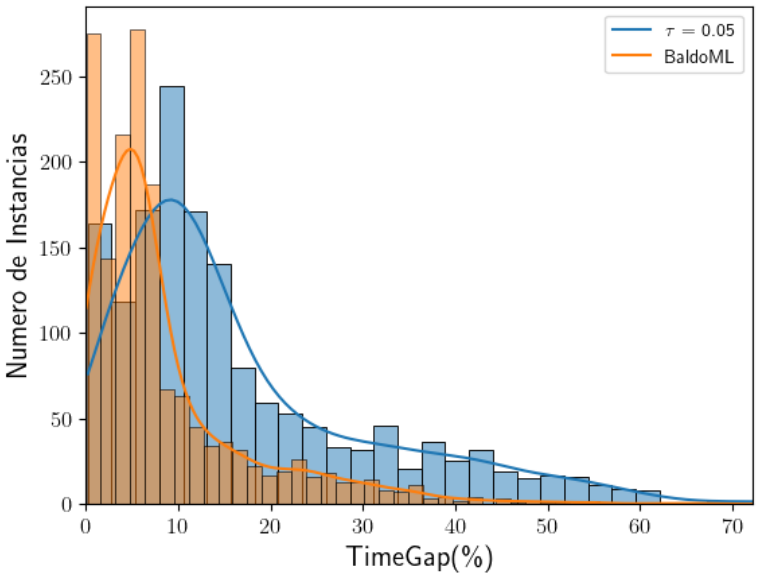

Figura 4: Gap Tiempo(%) de los métodos usados.

Tabla 4: Comparación general por número de elementos

Método			|I|							
			100	300	500	700	900	1100	1300	1500
BaldoGA	Gap(%)	μ	1.74	3.07	3.04	3.42	3.74	3.4	4.1	3.9
		σ	1.15	0.905	0.866	0.844	1.09	1.07	6.89	1.18
	Time(s)	μ	0.44	3.3	4.5	7.18	9.86	5.61	7.22	8.89
		σ	0.207	1.32	1.75	2.93	4.08	2.19	2.44	3.21
BaldoML	Gap(%)	μ	3.16	3.0	2.2	2.0	1.85	1.41	1.37	1.39
		σ	1.87	1.19	0.907	0.722	0.659	0.56	0.592	0.497
	Time(s)	μ	0.0988	0.36	0.48	0.791	1.19	1.5	2.12	2.85
		σ	0.0147	0.0577	0.119	0.277	0.47	0.881	1.33	1.67
$\tau = 0.05$	Gap(%)	μ	2.45	1.8	1.37	1.4	1.41	1.4	1.23	1.02
		σ	2.51	1.71	1.35	1.36	1.38	1.38	1.28	1.16
	Time(s)	μ	0.209	0.971	1.6	2.77	4.26	3.74	5.04	6.72
		σ	0.0997	0.392	1.17	2.07	3.27	2.97	3.72	4.72
$\tau = 0.1$	Gap(%)	μ	4.35	3.26	2.82	2.74	2.69	2.43	2.34	2.12
		σ	3.32	2.19	1.89	1.96	1.86	1.67	1.72	1.59
	Time(s)	μ	0.164	0.772	1.03	1.57	2.46	2.16	2.95	3.68
		σ	0.0658	0.249	0.547	0.879	1.89	1.58	2.31	2.85
$\tau = 0.15$	Gap(%)	μ	5.83	4.58	3.94	3.75	3.8	3.33	3.26	3.04
		σ	3.39	2.58	2.2	2.18	2.24	1.89	1.96	1.68
	Time(s)	μ	0.139	0.686	0.843	1.27	1.89	1.66	2.1	2.54
		σ	0.038	0.212	0.349	0.564	1.19	1.01	1.33	1.55
$\tau = 0.2$	Gap(%)	μ	6.94	5.92	4.9	4.58	4.7	4.11	4.05	3.96
		σ	3.69	2.91	2.33	2.24	2.42	2.05	2.03	1.85
	Time(s)	μ	0.131	0.622	0.744	1.1	1.53	1.39	1.69	2.03
		σ	0.0234	0.135	0.26	0.393	0.676	0.675	0.86	0.999
$\tau = 0.25$	Gap(%)	μ	7.84	6.84	5.76	5.42	5.54	4.81	4.84	4.84
		σ	3.95	3.04	2.37	2.28	2.46	2.15	2.07	1.99
	Time(s)	μ	0.126	0.596	0.685	0.974	1.32	1.2	1.44	1.68
		σ	0.0151	0.106	0.182	0.215	0.387	0.383	0.497	0.51

El método propuesto con $\tau = 0.05$ muestra un menor Gap en comparación con BaldoML y BaldoGA en todos los tamaños de instancias. Esto indica que el método propuesto tiene un mejor rendimiento en términos de calidad de la solución obtenida. Aunque tiene tiempos de ejecución ligeramente más altos en comparación con BaldoML, la diferencia podría considerarse moderada. En todos los casos la desviación estándar de los tiempos de ejecución del algoritmo propuesto aumenta con la complejidad del problema de forma más rápida que BaldoML. Las instancias de 100 elementos poseen gaps particularmente mas altos para el método propuesto y BaldoML, para luego estabilizarse en instancias más grandes.

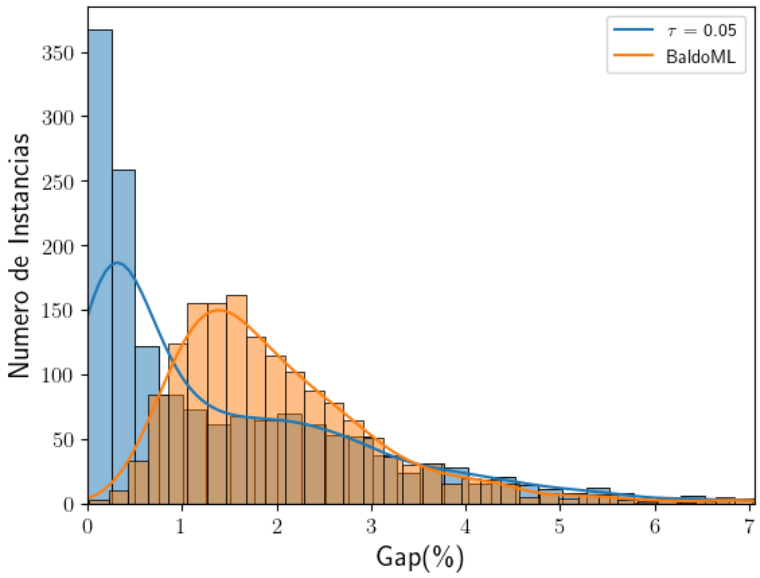

Figura 5: Contraste distribución gap BaldoML vs Propuesto

El histograma de la Figura 5 ilustra la principal diferencia entre BaldoML y el propuesto. Los métodos tienen un gap medio de 2.05 % y 1.51 % respectivamente, esta diferencia de 0.54 % es significativa con un $p = 1.8 \cdot 10^{-26} < 0.05$ en una prueba T-student, rechazando la hipótesis de igualdad de medias. El método propuesto concentra una mayor proporción de instancias en gaps inferiores al 1 % que BaldoML .

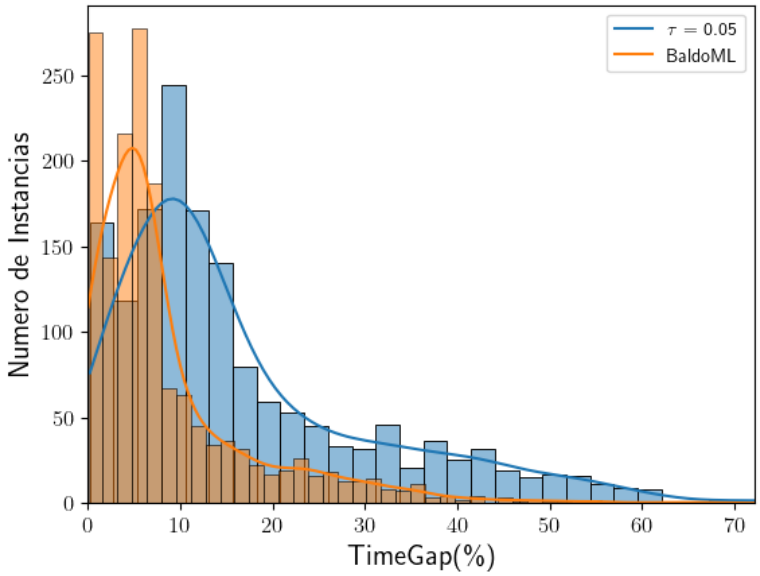

Figura 6: Contraste distribución tiempos de ejecución normalizados BaldoML vs Propuesto

Respecto a los tiempos de ejecución, BaldoML resolvió todas las instancias en 1859.86

segundos, lo que corresponde a 1.1719s en promedio por instancia. Mientras que el método propuesto tardó 4611.1s, con un promedio de 2.9 segundos. Debido a que las instancias tienen distintos tamaños y los tiempos de ejecución varían exponencialmente según este, se presenta en la Figura 6 una distribución de los tiempos normalizados. Ambas curvas se distribuyen de forma similar, con un desplazamiento positivo de la media del método propuesto, resultando en que una porción más alta de las instancias cae en TimeGaps de mayor magnitud. En promedio BaldoML se tarda 8.5 % del tiempo de gurobi en encontrar una solución, mientras que el propuesto un 15.93 %.

Instancias grandes

Debido a la inviabilidad de utilizar el solver exacto para estos tamaños, las 50 instancias de $|I| = 20.000$ se resolvieron usando los métodos BaldoML y el propuesto para ser comparados de forma directa. Además, en estos tamaños el clasificador demuestra ser inviable con un $\tau = 0,05$. Al enfrentarse a tamaños con los que no fue entrenado, entrega predicciones de menor confianza, es decir más lejanas a cero, resultando en que el algoritmo no elimina ningún elemento. Relajar τ a un valor de $0,25$ resuelve este problema satisfactoriamente.

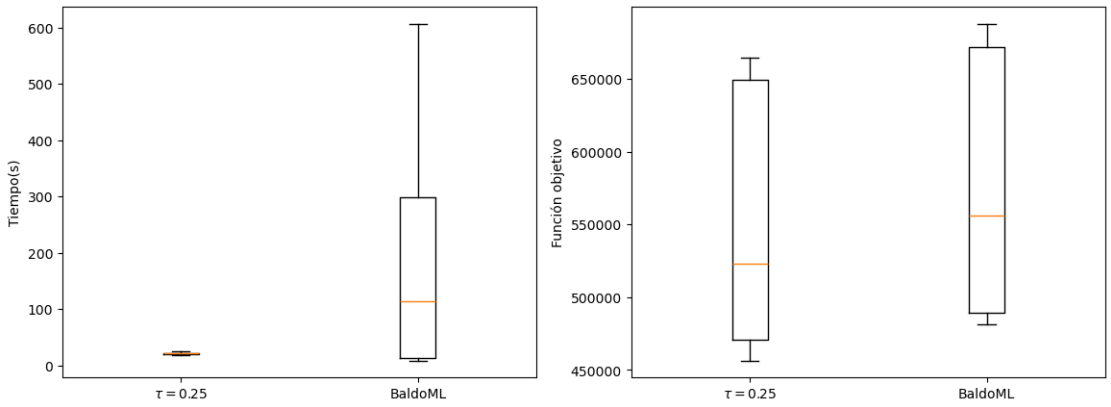

Figura 7: Comparación BaldoML y Propuesto en instancias de 20.000 elementos

En todos las instancias BaldoML encontró mejores soluciones, pero como se ilustra en la Figura 7 la diferencia entre esos es marginal. Las soluciones del método propuesto fueron en promedio un 4,47 % más pequeñas que las soluciones obtenidas por BaldoML, variando entre un 6,2 % y 2,85 % en el peor y mejor de los casos respectivamente. En cuanto a los tiempos de ejecución, la diferencia ha tenido mayor magnitud. Se estableció un límite de ejecución de 600 segundos por instancia. Mientras que BaldoML tardo 2 horas y 26 minutos en resolver las 50 instancias, el método propuesto usó poco menos de 18 minutos.

Su tiempo de ejecución se mantiene relativamente constante en estos tamaños, mientras los tiempos de BaldoML se distribuyen de forma más amplia, con una desviación estándar muy alta.

Tabla 5: Estadísticas, tiempos de ejecución en instancias grandes

	BaldoML	Propuesto
Tiempo total	8.809s	1.064,02s
Tiempo promedio	176,18s	21.28s
Desviación estándar	164,44s	1.60s

Capítulo 5

Conclusiones

Esta memoria de título propone un algoritmo basado en machine learning para abordar el problema polinomial robusto de la mochila, usando un clasificador que opera de forma iterativa sobre una instancia para reducir su complejidad. Se ha definido la estructura de la red neuronal, el algoritmo general de operación, los 2 métodos de entrenamiento y el método de reducción de la instancia.

En los resultados, el algoritmo propuesto muestra un rendimiento objetivamente superior al algoritmo genético propuesto por Baldo et al. (2023) en términos de precisión y tiempos de ejecución.Y Presenta mejores soluciones que BaldoML a pesar de tener tiempos de ejecución ligeramente mayores. Además, con una variación en los parámetros, tiene mejores características de escalamiento para abordar tamaños mayores a los contemplados en entrenamiento y evaluación. La metodología propuesta es entonces más adecuada para abordar instancias infactibles de resolver con un solver exacto, o con BaldoML en tiempos razonables.

En el futuro podría sería importante explorar el uso de técnicas más avanzadas de machine learning para abordar el problema. Implementar sistemas de decisión automática para los parámetros de los solvers mostrados y además explorar el uso de características derivadas de estimadores basados en machine learnig, como la arquitectura A2C introducida por Mnih et al. (2016), con el fin de predecir características sin necesidad de derivarlas de la solución óptima.

Bibliografía

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. Computing.*, 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.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783 [cs] version: 2.

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*.