Artículo Original / Original Research

Sinergia de Probabilidades, Optimización y Aprendizaje Automático para la Toma de Decisiones: Análisis de Resultados

Synergy of Probabilities, Optimization, and Machine Learning for Decision Making: Analysis of Results

Enzo Rojas D'Toste[1], Aymée de los Ángeles Marrero Severo[2]

Resumen

Este trabajo presenta los principales resultados de la estrategia formulada en un trabajo anterior [1]. Los casos de estudio incluyen un *juego de memoria con olvido*, un *entorno de apuestas con riesgo* y el problema del *cubrimiento mínimo de vértices* en grafos. Sobre cientos de instancias aleatorias, la estrategia jamás rinde peor que el algoritmo voraz (greedy) y mejora la solución, manteniendo complejidad temporal comparable.

En consecuencia, el marco probabilístico, combinado con optimización meta-heurística o neuronal, constituye una alternativa sólida y extensible para problemas combinatorios de decisión complejos.

Palabras Clave: toma de decisiones, asignación probabilística, metaheurísticas, Transformer, SVD.

Abstract

This work tackles the main results of the formulated strategy in a previous work [1]. Case studies include an *imperfect-memory matching game*, a *risk-based betting scenario*, and the *minimum vertex-cover* problem on graphs. Across hundreds of random instances, the proposed approach never underperforms a greedy baseline and improves solution quality, all while maintaining comparable computational complexity.

Hence, the probabilistic framework, coupled with meta-heuristic or neural optimisation, offers a robust and extensible alternative for difficult combinatorial decision problems.

Key words: decision making, probabilistic assignment, metaheuristics, Transformer, SVD.

Mathematics Subject Classification: 90B50, 90C59, 68W20, 68T05, 68T09.

[1]Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba. Email: enzord2001@gmail.com

[2]Departamento de Matemática, Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba. Email: aymee@matcom.uh.cu

Citar como: Rojas D'Toste E., Marrero Severo A. *Sinergia de Probabilidades, Optimización y Aprendizaje Automático para la Toma de Decisiones: Análisis de Resultados.* Ciencias Matemáticas, X(X), X–XX. Recuperado a partir de `https://revistas.uh.cu/rcm/`

1 Introducción

Este artículo constituye una *segunda parte* del trabajo, centrado exclusivamente en la presentación y el análisis de los **resultados experimentales**. La primera parte, descrita en [1], desarrolla los fundamentos conceptuales: definición de las funciones de asignación probabilística, procedimiento de ajuste mediante metaheurísticas [2] y extensión con *Transformers* [3] apoyados en reducción SVD (Singular Value Descomposition) [4].

Nuestros objetivos son:

1. **Validar empíricamente** las hipótesis teóricas, contrastando el rendimiento del algoritmo propuesto frente a *baselines* estándar (heurísticas *greedy*, metaheurísticas sin adaptación dinámica, etc.).
2. **Cuantificar** el impacto de cada componente—función probabilística, ajuste con PSO [5] y adaptación mediante *Transformer*—sobre distintas clases de problemas.

A lo largo del artículo se describen los protocolos de prueba, las métricas empleadas y los escenarios simulados o generados aleatoriamente, para luego discutir los resultados a la luz del marco teórico establecido en [1].

2 Pruebas y Resultados: Primer Caso de Estudio

En esta sección se presentan las pruebas realizadas para validar el desempeño del algoritmo en un juego de parejas de cartas con memoria imperfecta. El objetivo de este caso de estudio es ilustrar cómo el mecanismo de asignación probabilística, la optimización de parámetros y la selección dinámica mediante aprendizaje automático pueden adaptarse a un escenario donde la información no se conserva de manera perfecta.

2.1 Descripción del Juego y Motivación

El juego consiste en un conjunto de cartas boca abajo, dispuestas en posiciones fijas sobre la mesa. Cada carta tiene una pareja idéntica, también presente en algún lugar del tablero. La dinámica básica es:

1. En cada ronda, el jugador selecciona dos cartas para descubrirlas.
2. Si las cartas son iguales, se retiran del tablero (pareja encontrada).
3. Si son diferentes, se voltean nuevamente boca abajo.

El objetivo final es *quedarse sin cartas boca abajo*, es decir, descubrir todas las parejas.

2.1.1 Limitaciones de Memoria y Confusión

Para una máquina con memoria perfecta, el juego es trivial, ya que cualquier carta vista una vez puede recordarse con exactitud. Sin embargo, en el caso de los humanos, la *memoria no es perfecta*: a menudo se confunden las posiciones de las cartas descubiertas e, incluso, se puede olvidar por completo haber visto cierto naipe.

Con el fin de modelar esta limitación en la máquina, se introduce una *matriz de probabilidades* que contiene la información sobre dónde se cree que está cada carta, con cierto grado de incertidumbre. Específicamente:

- La matriz asocia a cada posición del tablero un vector de probabilidades, donde cada componente p_i indica la probabilidad de que *la carta i* se encuentre en esa posición.
- Cuando se descubren dos cartas en la ronda, si se obtiene información certera (dos cartas distintas u iguales), se actualiza la matriz para reflejar que, en las posiciones reveladas, la probabilidad de tener tal carta es 1, y la probabilidad de tener otras cartas es 0.
- Tras cada ronda, se aplica un mecanismo de **pérdida de memoria**, donde cierto α (porcentaje de olvido) redistribuye parte de la probabilidad asociada a cada carta entre las posiciones vecinas. De esta manera, se simula la *confusión* propia de la memoria imperfecta: mientras mayor sea α, más se propaga la incertidumbre al resto del tablero.

2.2 Estrategia de Decisión Probabilística

La estrategia básica para escoger qué dos posiciones revelar en cada ronda se basa en **valores esperados**. Para cada par de posiciones del tablero (x, y), se multiplican sus correspondientes vectores de probabilidad y se estima el valor esperado de la carta que se podría descubrir en ese par. A grandes rasgos:

- Si existe alta certeza de que dos posiciones contienen la misma carta, el valor esperado de elegirlas es mayor.
- Si la confusión (o pérdida de memoria) es muy elevada, estos valores esperados pueden volverse menos fiables.

El *algoritmo* que proponemos debe *aprender* cuándo es conveniente seguir los valores esperados más altos (que tienden a favor de las opciones aparentemente más seguras) o explorar otras posiciones cuya información se desconoce con mayor grado de incertidumbre.

2.3 Configuración de las Pruebas

Para llevar a cabo las pruebas, se define:

- **Tamaño del Tablero:** Se trabaja con un número de cartas equilibrado para que el problema sea desafiante pero manejable (10 pares).
- **Rango de Pérdida de Memoria (α):** Se evalúan escenarios con distintos niveles de confusión para medir la robustez de la estrategia.
- **Iteraciones Metaheurísticas:** La optimización se ejecuta bajo un límite de iteraciones con el fin de explorar soluciones factibles en un tiempo de cómputo razonable.
- **Entrenamiento de la Red Neuronal:** Se generan múltiples instancias simuladas con diferentes valores de α y composiciones de cartas.

2.4 Usando PSO

En esta sección se muestran los resultados obtenidos al aplicar el algoritmo de Enjambre de Partículas (*Particle Swarm Optimization*, PSO) para ajustar los parámetros de la función logarítmica invertida de asignación probabilística descrita en [1]. El objetivo es minimizar el número de pasos necesarios en promedio para resolver el juego de parejas con un cierto nivel de confusión (α), teniendo en cuenta que **existe un límite de rondas** máximo para no permitir partidas excesivamente largas.

2.4.1 Función Objetivo del PSO

Para cuantificar la calidad (o aptitud) de una configuración de parámetros, se define una **función objetivo** basada en:

- **Cantidad de pasos** promedio necesarios para resolver el juego cuando el número de rondas totales *no* supera un cierto límite (limit).
- **Partidas fallidas o sobre el límite**, es decir, aquellas que exceden la cantidad de pasos permitidos, lo cual podría considerarse un fracaso en la práctica.

Se definen:

- count: la cantidad total de partidas ejecutadas durante la evaluación.
- ol (*over limit*): la cantidad de partidas que superaron el límite de pasos permitido.
- total: la suma de los pasos utilizados en las partidas que finalizaron por debajo del límite.

Entonces, la función objetivo (o **fitness**) se define como:

$$\text{fitness} = \begin{cases} \text{ol} + 2, & \text{si count} = \text{ol}, \\ \text{ol} + 2 \times \left(\frac{\text{total}}{\text{count} - \text{ol}}\right)/\text{limit}, & \text{en caso contrario.} \end{cases}$$

La expresión anterior se interpreta de la siguiente manera:

- Si todas las partidas (count = ol) se fueron por encima del límite, el algoritmo retorna ol+2 como penalización máxima, evidenciando un desempeño muy deficiente.

- De lo contrario, se penaliza el número de partidas fallidas (ol) y, además, se considera el promedio de pasos (total/(count − ol)), escalado por limit.

2.4.2 Configuración Experimental

Para llevar a cabo los experimentos se fija:

- **Número de partículas** en el enjambre: parámetro usual de PSO (por ejemplo, 20 o 30 partículas).
- **Iteraciones máximas** de PSO: determinado para equilibrar la exploración y el tiempo computacional.
- **Niveles de confusión** (α): se prueban valores representativos (por ejemplo, 0.1, 0.3, 0.5, etc.) para cubrir escenarios de memoria más confiable y otros sumamente confundidos.

2.4.3 Visualización de la Función Logarítmica Invertida con Parámetros Obtenidos

Con el fin de ilustrar cómo varía la *función logarítmica invertida* (empleada en la asignación probabilística) al ajustar los parámetros encontrados por PSO, se muestra el caso particular con $n = 5$. Esto corresponde a un escenario donde existen 5 decisiones potenciales (o parejas) por elegir, lo que brinda una representación gráfica clara de cómo la probabilidad se distribuye entre ellas. En la práctica del juego, n varía a medida que se retiran parejas del tablero, por lo que estos porcentajes se adaptan dinámicamente, aunque la forma cualitativa de las curvas se conserva.

A continuación, se presentan distintas gráficas que muestran la forma de la función logarítmica invertida para diferentes valores de α (porcentaje de confusión):

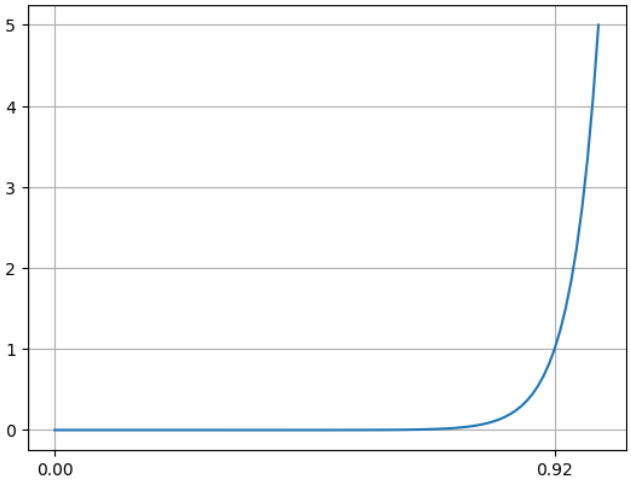

Figura 1: Curva logarítmica invertida para α entre 0 % y 2 %. Se observa que la primera opción recibe en promedio un 92 % de confianza.

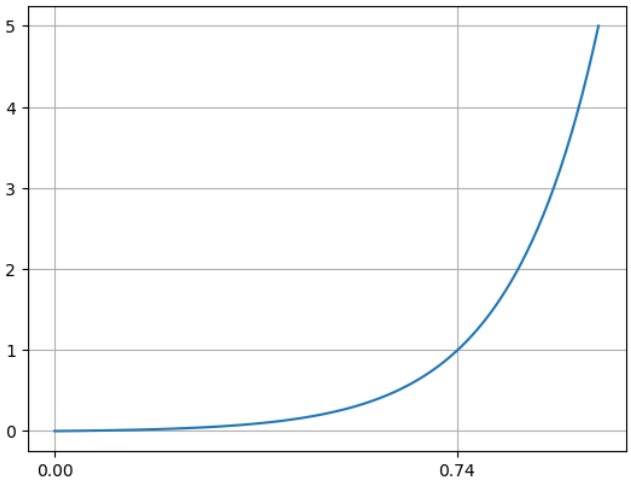

Figura 2: Curva logarítmica invertida para $\alpha =$ 5 %. La primera opción obtiene un 74 % de confianza.

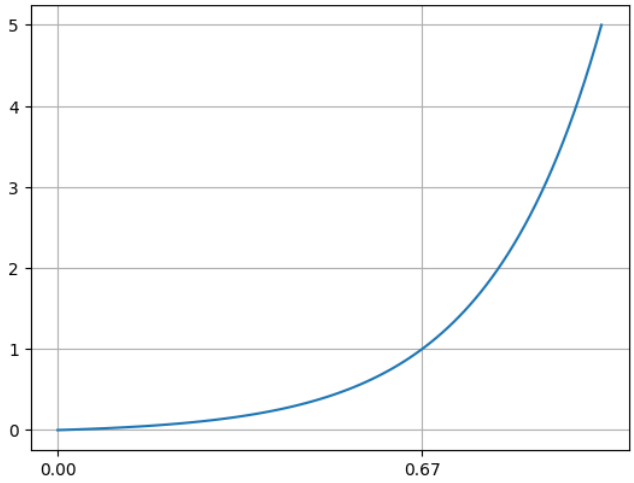

Figura 3: Curva logarítmica invertida para $\alpha = 10\,\%$. La primera opción baja a un $67\,\%$ de confianza, indicando más distribución hacia las demás.

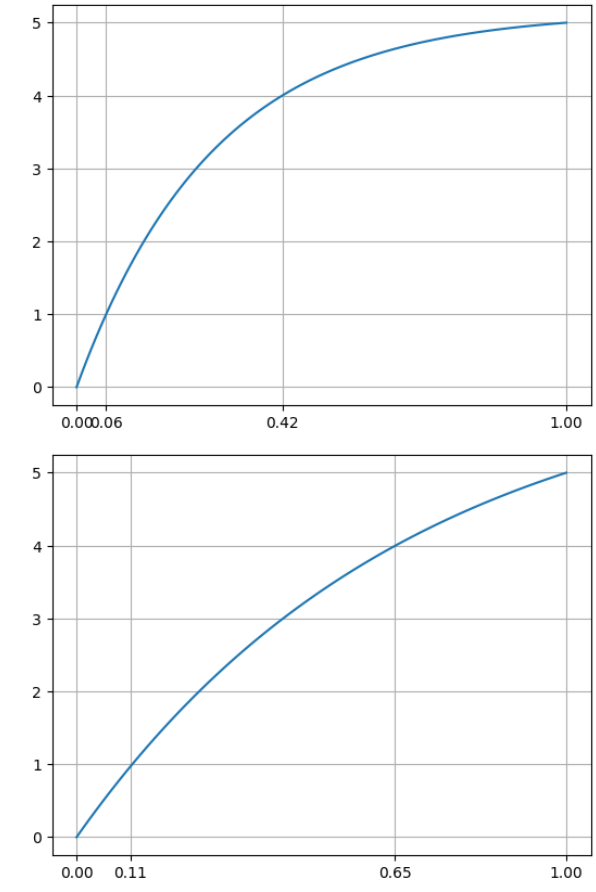

Figura 4: Curvas logarítmicas invertidas para α entre $50\,\%$ y $80\,\%$. Se observa que la primera opción cae entre $6\,\%$ y $11\,\%$, mientras que la última asciende entre $35\,\%$ y $58\,\%$.

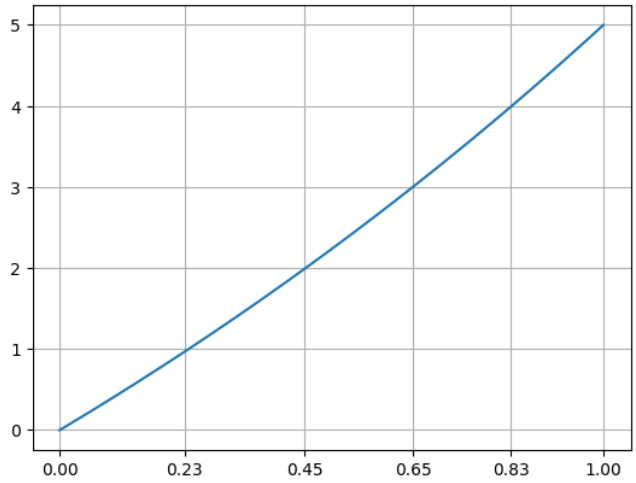

Figura 5: Curva logarítmica invertida para $\alpha = 100\,\%$. Todas las opciones reciben aproximadamente la misma probabilidad (alrededor de un $20\,\%$ para $n = 5$).

Se aprecia cómo, a medida que incrementa α, la función resultante va reduciendo la confianza en la primera decisión y distribuyéndola cada vez más entre el resto de las opciones. De hecho, para niveles de confusión muy altos ($50\,\%$–$80\,\%$), se observa un comportamiento inverso, favoreciendo las opciones finales en mayor proporción. Finalmente, con $\alpha = 100\,\%$, las probabilidades quedan *uniformemente* repartidas, dado que la información se considera completamente confusa.

Estos resultados muestran cómo los parámetros determinados por PSO ajustan la función logarítmica invertida de forma adaptativa, favoreciendo la explotación cuando la memorización del juego es confiable (baja confusión) y diversificando la búsqueda de parejas cuando la memoria es sumamente inestable (alta confusión).

2.4.4 Comparación de Resultados con la Selección por Mayor Valor Esperado

Para evaluar la eficacia de los parámetros encontrados mediante PSO (en conjunto con la función logarítmica) se compara su desempeño con el de una estrategia que siempre elige la decisión con mayor valor esperado. Se simularon 100 partidas para cada nivel de confusión (α), con un límite de hasta 1000 rondas. En caso de superar ese número de rondas, consideramos que la partida entra en

un ciclo indefinido y, por tanto, la categorizamos como un fallo (*over limit*).

En la tabla siguiente se muestran, para cada valor de α:

- **Promedio debajo del límite** (sólo considerando las partidas que no superaron las 1000 rondas).
- **Cantidad de partidas por encima del límite** (las que potencialmente se vuelven infinitas).

Cuadro 1: Comparación de la media de rondas y fallos entre el algoritmo propuesto (PSO + función logarítmica) y la selección por mayor valor esperado. Cada celda muestra (Promedio, # over limit).

α	Algoritmo (PSO)	Mayor Criterio
0.0	(21.52, 0)	(21.53, 0)
0.01	(24.95, 0)	(25.38, 0)
0.02	(25.14, 0)	(26.65, 0)
0.05	(30.74, 0)	(36.08, 0)
0.1	(60.43, 0)	(170.67, 2)
0.3	(120.67, 0)	(443.11, 17)
0.5	(108.49, 0)	(474.96, 20)
0.8	(100.22, 0)	(429.92, 29)
1.0	(98.65, 0)	(414.17, 40)

Análisis de Resultados

- **Escenario sin confusión** ($\alpha = 0$): Ambas tienen un rendimiento muy parecido y sin partidas fallidas.
- **Confusión baja a moderada** ($0{,}01 \leq \alpha \leq 0{,}3$):
  - El algoritmo propuesto mantiene su capacidad de resolver todas las partidas por debajo del límite, con promedios crecientes a medida que α sube, pero sin llegar a fallo.
  - La estrategia de mayor valor esperado, en cambio, comienza a acumular partidas fallidas; a partir de $\alpha = 0{,}1$ aparecen 2 fallos, esta tendencia se agrava a 17 fallos con $\alpha = 0{,}3$.
- **Confusión alta** ($0{,}5 \leq \alpha < 1{,}0$):
  - El método con PSO sigue sin generar ninguna partida fallida, con promedios entre 100 y 108 rondas.
  - La selección por mayor valor esperado se vuelve inviable, con un número considerable de partidas (entre 20 y 29 de cada 100) rebasando el límite. Incluso el promedio entre las partidas que no fallan es muy alto (por ejemplo, 474.96 rondas para $\alpha = 0{,}5$).
- **Confusión total** ($\alpha = 1{,}0$):
  - El algoritmo propuesto resuelve la totalidad de las partidas sin fallar, con un promedio de 98.65 rondas.
  - La estrategia de mayor valor esperado falla en una gran cantidad de las partidas, y con un alto promedio de rondas en el resto.

Estos datos reflejan que, para $\alpha = 0$ el algoritmo decide que la selección por mayor valor esperado es lo más eficiente, pero **en cuanto existe un mínimo grado de confusión la estrategia de mayor valor esperado deja de ser confiable** y provoca ciclos infinitos en una fracción importante de las partidas. Por el contrario, el algoritmo que combina PSO con la función logarítmica invertida y un método probabilístico de selección consigue adaptarse a la pérdida de información y culminar exitosamente el juego incluso para altos valores de α.

Conclusión preliminar: La capacidad de explorar nuevas decisiones en lugar de confiar exclusivamente en la *opción aparentemente más prometedora* resulta crítica en escenarios donde la información puede ser parcial o confusa.

2.5 Usando Transformer

En la sección 2.4, se mostró cómo, mediante metaheurísticas como PSO, es posible encontrar conjuntos de parámetros que optimizan el desempeño del juego de parejas para un *nivel de confusión* (α) en particular. Sin embargo, en la práctica, **no siempre conocemos el valor de α con exactitud**. Por ello, el siguiente paso consiste en incorporar el componente de **aprendizaje automático** (redes neuronales) para que el algoritmo se *adapte* dinámicamente a la situación real del juego, infiriendo la confusión a partir de los estados.

2.5.1 Entrenamiento y Construcción del Conjunto de Datos

Para implementar el componente de **Transformers** se siguen las adaptaciones explicadas en secciones anteriores, incluyendo:

- El uso de **SVD** para asegurar la *escalabilidad* cuando el número de estados o la longitud de la secuencia aumenta significativamente.
- La *codificación* de los estados del juego de parejas, representando la matriz de probabilidades, las cartas eliminadas y otras características relevantes.

El **dataset de entrenamiento** se compone de *instancias específicas* del juego (misma disposición de cartas y confusión), para las cuales se obtienen los *parámetros óptimos* usando PSO. A diferencia de la estrategia pura de PSO, donde se busca un conjunto de parámetros que funcionen *en promedio* para mútiples casos de una misma confusión, aquí se recolectan soluciones *individualizadas* (una por cada instancia), enriqueciendo los ejemplos con escenarios más específicos y precisos.

Para fines demostrativos y por *limitaciones de tiempo*, se generó un **conjunto de datos relativamente pequeño** y se entrenó el Transformer en *pocos épocas* (epochs). Aún así, los resultados obtenidos fueron prometedores, sugiriendo que un entrenamiento más extenso y con más casos podría mejorar significativamente el desempeño del modelo.

2.5.2 Comparativa con PSO: Inferencia Dinámica de la Confusión

A continuación, se realizó una prueba cubriendo valores de α en el rango $[0, 1]$. La estrategia basada **exclusivamente** en PSO emplea un *conjunto discreto* de soluciones (las que optimizan cada α discreto considerado), y en tiempo de decisión elige el conjunto de parámetros cuyo α entrenado sea más cercano al valor real. El **Transformer**, en cambio, produce un *conjunto continuo* de parámetros y *no* requiere conocer α, pues *infiere* el nivel de confusión implícitamente a partir de la información de estado.

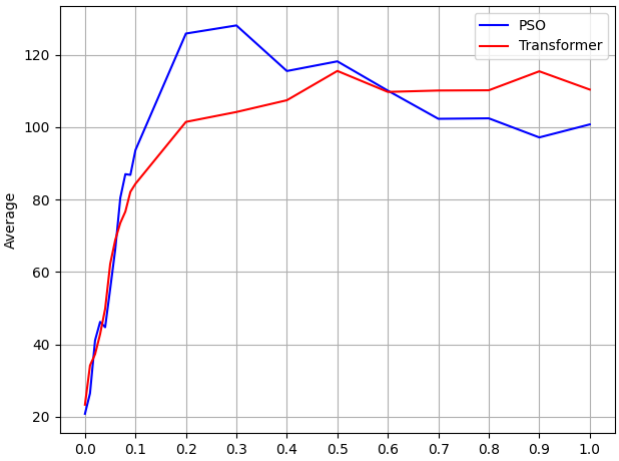

Figura 6: Comparación de resultados entre el Transformer y la estrategia puramente basada en PSO, para valores de α entre 0 y 1.

En la Figura 6, se aprecia la evolución de los resultados de ambos métodos al variar α. Ninguno de los dos generó partidas por encima del límite, lo cual es una buena señal de la robustez de ambas propuestas. Al desglosar el análisis:

- **Confusión hasta 10 %**: Los resultados de PSO y Transformer son muy similares. PSO tiende a mostrar un ligero dominio en α más bajos, posiblemente por haber encontrado parámetros muy precisos para ese entorno estable.
- **Confusión entre 10 % y 60 %**: El *Transformer* se convierte en un claro ganador, lo que evidencia el **poder de tener parámetros específicos para cada instancia** y la capacidad de adaptación dinámica. El PSO pierde eficiencia al intentar cubrir un espectro genérico de confusiones, mientras que el Transformer aprovecha su entrenamiento instancia a instancia.
- **Confusión entre 60 % y 100 %**: El PSO recupera cierta ventaja, superando ligeramente al Transformer. Aun así, los desempeños son de nuevo cercanos, pues ambos muestran una monotonía similar a medida que se acerca la confusión total.

Esto indica que **un Transformer con un conjunto de entrenamiento más amplio, abar-**

cando múltiples instancias y confusiones variadas, podría superar en la mayoría de los casos la estrategia basada exclusivamente en PSO, además de su ventaja inherente de *no requerir el conocimiento previo* de α.

3 Aplicación en un Entorno de Riesgo

En esta sección se presenta otra prueba del algoritmo, enfocada en un **problema de ganancia bajo riesgo**. Se tienen n rondas, y en cada ronda el agente decide cuánto apostar ($\in [0, +\infty)$). Existe una *función de riesgo*, simulada en este caso por una probabilidad desconocida, que determina si la cantidad apostada se *suma* o *resta* al capital acumulado. El objetivo final es **maximizar el capital** tras completarse las n rondas.

3.1 Modelo y Ajuste Mediante Proporcionalidad Inversa

Dado que el valor a apostar puede ser arbitrariamente grande, se opta por la **función de proporcionalidad inversa** como mecanismo de asignación probabilística, asegurando que para valores de entrada $x \in [0, 1)$ se mapee a $[0, +\infty)$. Posteriormente, se emplea **PSO** para encontrar los parámetros de dicha función que maximizan la ganancia promedio en un *entorno de riesgo* dado por:

$$\text{Probabilidad de Ganar/Apostar} \;=\; \alpha\,|\sin(i)|,$$

donde i es el número de ronda, y α varía en el rango [0, 1]. La idea es que, cuanto mayor sea α, *más volátil* es el ambiente de apuestas, incrementando el riesgo de pérdida.

3.1.1 Pruebas y Resultados con PSO

Para evaluar la efectividad de los parámetros de la función de proporcionalidad inversa, se realizaron 100 ejecuciones del juego, ajustando la duración a un número fijo de rondas y registrando:

- **Promedio de ganancia**: capital final promedio tras las 100 ejecuciones.
- **Cantidad de veces que se pierde dinero**: número de ejecuciones (de 100) en las que el capital final resulta menor que 0.

La *función objetivo* usada por PSO se definió para equilibrar la maximización de la ganancia y la reducción del número de pérdidas. A continuación, se presentan los resultados obtenidos para distintas configuraciones de α.

Cuadro 2: Resultados al variar α en el rango $[0, 1]$. Cada valor se basa en 100 ejecuciones.

α	Perdidas	Ganancia Promedio
0.0	0	1.05×10^{15}
0.1	0	2.13×10^{15}
0.2	3	3.22×10^{14}
0.3	3	5.74×10^{14}
0.4	5	9.59×10^{14}
0.5	18	6.81×10^{14}
0.6	18	4.06×10^{14}
0.7	33	5.05×10^{14}
0.8	52	2.76×10^{14}
0.9	65	3.83×10^{14}
1.0	83	7.57×10^{14}

Análisis de los Resultados

- $\alpha = 0{,}0$ y $\alpha = 0{,}1$: Se obtienen ganancias extremadamente altas, con **cero pérdidas** en 100 ejecuciones. El entorno es, en práctica, muy estable, lo que facilita el incremento sostenido del capital.
- $\alpha = 0{,}2$ a $\alpha = 0{,}4$: Comienza a existir cierto *riesgo*, reflejado en unas pocas ejecuciones con pérdidas, mientras los valores promedio de ganancia siguen siendo del orden de 10^{14} o 10^{15}.
- $\alpha \geq 0{,}5$: A medida que sube la volatilidad, aumenta notablemente la probabilidad de pérdida, llegando a más de la mitad de las ejecuciones para $\alpha = 0{,}8$ y superiores. Sin embargo, el capital promedio sigue resultando muy alto, indicando que en las partidas favorables se logran ganancias enormes, compensando en parte las caídas.

Estos resultados evidencian que, si bien se pueden alcanzar ganancias elevadas incluso en entornos

muy volátiles, *el precio a pagar es un incremento sustancial en la posibilidad de terminar en números rojos*. Esto es consistente con la idea de que, al usar la función de proporcionalidad inversa, se pueden tomar apuestas más elevadas en condiciones de alto riesgo, lo que dispara tanto la ganancia como la pérdida potencial.

3.2 Pruebas con la Inversa de la Exponencial

Además de la función de proporcionalidad inversa, se consideró una **función exponencial inversa** para la selección del valor a apostar. La expresión, partiendo de un valor $x \in [0, 1)$, se define como:

$$f(x) \;=\; -\lambda \,\ln\big(1 - x\big),$$

donde $\lambda > 0$ es el parámetro a optimizar. Nótese que:

- Para $x \approx 0$, $f(x) \approx 0$, de modo que se opta por apuestas conservadoras.
- A medida que x se acerca a 1, $-\ln(1 - x)$ tiende a $+\infty$, haciendo crecer $f(x)$ sin cota superior, lo que sigue permitiendo apuestas elevadas.
- El valor λ controla la *escala* de la apuesta, similar a la media deseada en la distribución exponencial.

3.2.1 Configuración Experimental y Resultados

Se mantiene el mismo escenario de n rondas con una función de riesgo desconocida, modelada por $\alpha \sin(i)$ para cada ronda i. Se realizaron 100 ejecuciones para cada valor de $\alpha \in \{0; 0{,}1; 0{,}2; 0{,}3; 0{,}4; 0{,}5; 0{,}6; 0{,}7; 0{,}8; 0{,}9; 1{,}0\}$. El **PSO** se encarga de encontrar el λ que maximice la ganancia promedio y minimice el número de pérdidas.

A continuación se presentan los resultados, mostrando (en 100 ejecuciones):

- **Veces que se pierde dinero** (capital final menor que 0).
- **Promedio de ganancia** entre todas las partidas jugadas.

Cuadro 3: Resultados con la función inversa de la exponencial para distintos valores de α. Cada valor corresponde a (Veces que se pierde dinero, Promedio de ganancia).

α	Perdidas	Ganancia Promedio
0.0	0	6633.87
0.1	0	3748.56
0.2	0	9675.27
0.3	0	4645.20
0.4	0	3673.89
0.5	0	3582.46
0.6	0	1953.96
0.7	1	727.12
0.8	61	31.89
0.9	28	0.32
1.0	8	0.05

3.2.2 Análisis de Resultados

- **Rangos bajos de α (0 a 0,6):** Se observa un número de pérdidas cero, mientras que las ganancias promedio varían. El comportamiento es más *conservador* que con la función de proporcionalidad inversa, pues no se realizan apuestas desmesuradas: esto se refleja en ganancias más moderadas, pero también en menores riesgos.
- **Riesgo mayor ($\alpha = 0{,}7$ a 1,0):** Empiezan a surgir pérdidas más frecuentes. El promedio de ganancia baja a niveles cercanos a cero. Esto indica que la inversa exponencial se vuelve menos lucrativa a medida que la volatilidad del riesgo aumenta, aun siendo más "estable" que la otra función.
- **Comparación con la Proporcionalidad Inversa:** Mientras la función logarítmica inversa puede generar ganancias enormes a costa de un *alto riesgo*, la inversa exponencial resulta más conservadora en sus apuestas, ofreciendo valores más consistentes y reduciendo considerablemente las ocasiones en que se pierde dinero.

En conclusión, **la función exponencial inversa** constituye una opción más moderada para entornos muy volátiles, reduciendo el número de pérdidas incluso cuando α se torna elevado, a expensas de

no alcanzar *picos de ganancias* tan impactantes como la solución más agresiva. Este compromiso entre potencial de ganancia y estabilidad es clave al diseñar estrategias de decisión en entornos de riesgo.

3.3 Combinando ambas Funciones

Dado que cada función propuesta, la *proporcionalidad inversa* y la *inversa exponencial*, presenta fortalezas y debilidades para distintos niveles de riesgo, surge la idea de combinar múltiples opciones bajo una *función maestra*. En esta prueba, se utilizó la *función logarítmica inversa* como la encargada de asignar probabilidades a cada una de las dos funciones candidatas:

- **Proporcionalidad inversa**, cuya capacidad de generar ganancias masivas se ve compensada por un riesgo más alto.
- **Inversa exponencial**, que mantiene un comportamiento más conservador, reduciendo las pérdidas a costa de no alcanzar picos de ganancia tan elevados.

La **función maestra** determina, en cada ronda, la probabilidad de elegir una u otra estrategia para fijar la apuesta. Posteriormente, la función seleccionada asigna el valor final de la decisión según sus propios parámetros. De este modo, el algoritmo puede adaptarse dinámicamente conforme varía el entorno de riesgo.

3.3.1 Resultados y Análisis

Para comprobar la eficacia de esta combinación, se llevaron a cabo experimentos con distintos valores de α, manteniendo la misma función de riesgo $\alpha \sin(i)$. A continuación, se muestran (en 100 ejecuciones) la cantidad de partidas con pérdidas y la ganancia promedio:

Cuadro 4: Resultados al combinar la proporcionalidad inversa y la inversa exponencial, usando una función logarítmica inversa como maestra. Los datos corresponden a (Veces que se pierde dinero, Promedio de ganancia) en 100 ejecuciones.

α	**Pérdidas**	**Ganancia Promedio**
0.0	0	3.60×10^{14}
0.1	0	4.01×10^{14}
0.2	0	4.02×10^{15}
0.3	6	1.37×10^{14}
0.4	5	3.07×10^{14}
0.5	10	8.29×10^{13}
0.6	0	2490.56
0.7	0	1978.15
0.8	51	1.47×10^{13}
0.9	1	0.02
1.0	2	0.25

Observaciones Principales:

- $\alpha \leq 0{,}5$**:** Para riesgos bajos o moderados, la estrategia *tiende a seleccionar* con mayor frecuencia la *proporcionalidad inversa*, pues el número de pérdidas es relativamente bajo y los incrementos de capital pueden ser muy elevados. En estas condiciones, el algoritmo explota las grandes ganancias potenciales.
- $\alpha = 0{,}6$ **y** $\alpha = 0{,}7$**:** Cuando la volatilidad incrementa, la *proporcionalidad inversa* empieza a mostrar más pérdidas. La función maestra, entonces, **decide favorecer la inversa exponencial**, que garantiza un número de pérdidas cercano a cero a cambio de ganancias más moderadas.
- $\alpha = 0{,}8$**:** En este valor intermedio-alto, la *inversa exponencial* también exhibe problemas (mayores pérdidas y menor ganancia), de modo que la estrategia vuelve a inclinarse hacia la *proporcionalidad inversa*, a pesar del riesgo, esperando aprovechar rachas favorables para compensar las pérdidas.
- α **muy alta (**$0{,}9$ **y** $1{,}0$**):** Ambas funciones pierden efectividad; el algoritmo opta, en la práctica, por *apostar valores muy próximos a 0* con alta probabilidad, reduciendo así el riesgo de pérdida. Esto se refleja en las cifras:

el promedio de ganancia es casi despreciable, pero también se controla el número de partidas negativas.

Conclusiones:

- La **función maestra** resulta una excelente extensión, pues permite combinar *lo mejor* de múltiples estrategias en respuesta al *nivel de riesgo real* que se va detectando.
- En *riesgos bajos*, apostar más agresivamente (proporcionalidad inversa) maximiza ganancias; conforme la volatilidad aumenta, es preferible la inversa exponencial; y en casos extremos de altísimo riesgo, lo mejor es escoger *apuestas cercanas a cero*.
- Usando la estrategia del Transformer este enfoque podría ajustarse aún más finamente, *infiriendo dinámicamente* la situación de riesgo y adaptando la decisión en cada ronda.

De esta forma, la **combinación de funciones probabilísticas, gestionada por una función maestra**, pone de manifiesto cómo el algoritmo puede *evolucionar* según el nivel de riesgo presente en el entorno, sin limitarse a una sola estrategia que podría fallar rotundamente en determinadas circunstancias.

4 Mínimo Conjunto de Vértices de Cobertura

El **problema de encontrar la mínima cantidad de vértices de un grafo cuyas aristas alcancen a todos los nodos** (Mínimo Conjunto de Vértices de Cobertura, o *Minimum Vertex Cover*), es **NP-Completo**, implicando que, al crecer el número de nodos, encontrar la solución óptima en tiempo polinómico se considera inalcanzable a menos que P = NP. Por ello, se usan *estrategias aproximadas* que potencialmente devuelven valores cercanos al óptimo, aunque pueden excederlo.

4.1 Estrategia Aproximada Greedy

Una estrategia común de aproximación es:

1. Ordenar los nodos por su número de aristas (grado).
2. Escoger el nodo con **mayor grado**.
3. Remover ese nodo y todos los nodos que alcanza a *distancia de una arista* (junto a sus aristas correspondientes).
4. Repetir en el *grafo resultante* hasta cubrir todos los nodos.

En grafos donde es posible calcular el óptimo (digamos, de hasta 50 nodos), se observa que este método *casi siempre* retorna el valor óptimo o (óptimo + 1). Sin embargo, que sea muy efectivo *no* significa que no haya margen de mejora. Si se traslada a un problema logístico (por ejemplo, localización de centros de datos para cubrir todas las regiones de un país), un centro adicional podría representar *millones de dólares*.

4.2 Aplicando la Solución Propuesta

La idea es que en cada ronda, en lugar de *forzar* el nodo de mayor grado, se introduzca **exploración** gracias a la **función de asignación probabilística**, pudiendo descubrir combinaciones que el greedy omitiría.

4.3 Estrategia Conjunta: Mezclando el Greedy y el Algoritmo Probabilístico

El **método aproximado** descrito (ordenar por número de aristas y seleccionar el nodo con mayor grado) resulta muy eficiente, aunque no garantiza siempre la solución óptima. De hecho, puede haber *empates* entre varios nodos de máximo grado, y aun así el método no exploraría opciones con grado algo menor que podrían, combinadas con otros nodos, *cubrir* al grafo con menos nodos.

Incorporando la Función de Asignación Probabilística

- **Decisiones en cada ronda**: se dispone de un conjunto de nodos candidatos (los del grafo aún no cubiertos).
- **Criterio original**: escoger *siempre* el nodo de mayor grado.

- **Función de Asignación (FA)**: en vez de forzar el nodo de máximo grado, se asigna a cada nodo una probabilidad proporcional a su grado; a su vez, existe cierta *probabilidad no nula* de escoger un nodo que no sea el máximo. Esto introduce **exploración**, pudiendo así descubrir coberturas no alcanzadas por el greedy puro.

Estrategia Conjunta

- **Ajustar Parámetros de FA en general**: si se entrenan los parámetros para *todas* las instancias posibles, es probable que la estrategia resultante se parezca al greedy (dado lo efectivo que es en la mayoría de casos).
- **Enfoque Específico en los "casos difíciles"**:
  - Se generan n grafos y se compara *greedy* vs. óptimo.
  - Sólo en *los casos* donde el greedy no alcanza el óptimo se enfoca la optimización de la estrategia probabilística.
- **Combinación Final**:
  1. Para cada grafo, se corre el *greedy* y el *algoritmo probabilístico*.
  2. Se elige la *cobertura mínima* entre ambas salidas.

  Esta combinación garantiza no ser peor que el greedy, y *puede* superarlo en ciertos escenarios.

Complejidad y Práctica

- **Complejidad similar al Greedy**: El algoritmo base mantiene la misma complejidad temporal de la heurística (ordenar nodos, etc.).
- **Constante Adicional**: Surge al introducir la FA, sobre todo si se aplica un *Transformer* con SVD. Aunque esto añada un coste computacional, es todavía *polinómico* y, en la práctica, sólo un factor multiplicativo.
- **Beneficio**: Nunca empeora el resultado del greedy y, en casos donde el greedy falla, *logra mejoras* gracias a la exploración probabilística.

4.4 Pruebas con Grafos Aleatorios de Hasta 50 Nodos

4.4.1 Generando y Comparando con el Óptimo

- Se generaron 400 grafos con ≤ 50 nodos, probabilidad 50 % de arista entre cada par de nodos.
- Para cada grafo, se calculó el óptimo (vía algoritmo exhaustivo).
- El greedy resultó no óptimo en 44 grafos, confirmando que *puede fallar*.

4.4.2 Uso de la Función Logarítmica Inversa

Dado que en cada ronda tenemos un conjunto finito de decisiones (nodos), y se quiere favorecer con *mayor probabilidad* al de grado alto, se escoge la **función logarítmica inversa**. Para ajustar sus parámetros, se usa **PSO** con la siguiente función de *fitness* (pseudocódigo):

```
def fitness(graph):
    worst_case = None
    for _ in range(10):
        v = problem.run(graph)
        if worst_case is None
                      or worst_case < v:
            worst_case = v
    return worst_case
```

Resultados:

- Para cada uno de los 44 grafos donde el greedy fallaba se optimizaron los parámetros de la función, y se obtuvo que la solución *empató* al greedy en 34 y *ganó* en 10, *sin perder en ninguno*.

4.5 Entrenamiento de un Transformer

- Se entrenó el Transformer (con la técnica de SVD para entradas de estado) en **los 10 casos** donde se venció al greedy.
- Posteriormente, se generaron 100 nuevos grafos (hasta 50 nodos, probabilidad 50 % de arista).
- *Greedy* acierta el óptimo en 89 casos, mientras la *estrategia conjunta* llega a 93.
- Si bien la mejora es discreta, se acerca un poco más al valor perfecto de 100 % de casos óptimos, además de que en problemas donde *un solo nodo extra* implica millones de dólares, cualquier avance es valioso.

4.6 Conclusiones

- El **greedy** es muy eficaz en grafos aleatorios de tamaño moderado, pero no siempre logra el óptimo.
- La estrategia probabilística, combinada con una metaheurística (PSO) o un modelo neuronal (Transformer), mejora *ligeramente* en los casos donde el greedy falla.
- La *combinación* final: tomar la *solución mínima* entre {greedy, algoritmo probabilístico} garantiza que nunca empeora los resultados, y potencialmente gana en ciertos grafos difíciles.
- Entrenar con más instancias y permitir arquitecturas de *mayor complejidad* (más capas neuronales) podría incrementar el número de casos donde se supera el greedy.

En definitiva, aunque el **ganar 4 casos más** en 100 grafos *parezca* poco, en un problema logístico a gran escala, *un solo nodo menos* puede equivaler a un ahorro masivo en recursos, justificando plenamente el uso de estrategias probabilísticas adaptativas.

5 Conclusiones Generales

A lo largo de este trabajo se validó un **algoritmo híbrido** que integra la asignación probabilística de decisiones, la optimización mediante metaheurísticas (PSO) y el uso de redes neuronales (Transformer) para abordar diversos problemas de toma de decisión bajo incertidumbre. Se presentan a continuación las conclusiones más destacadas:

1. **Asignación Probabilística:**
   - El uso de funciones tales como la *logarítmica inversa*, la *proporcionalidad inversa* o la *inversa exponencial* permite convertir un valor aleatorio uniforme en un índice (o cantidad) que asigna probabilidades a las decisiones disponibles.
   - Se demostró que, mediante condiciones de borde y normalización, es posible garantizar que la asignación cumpla las propiedades deseadas, favoreciendo la exploración y explotando las mejores opciones.
2. **Optimización por Metaheurísticas (PSO):**
   - La selección de parámetros óptimos para dichas funciones probabilísticas mediante PSO mostró resultados sólidos en diversos problemas, desde la *memorización imperfecta* hasta entornos de *riesgo* con funciones desconocidas.
   - La capacidad de PSO para explorar el espacio de parámetros de manera estocástica y eficiente garantiza que, con un número moderado de iteraciones, se encuentren configuraciones competitivas.
3. **Redes Neuronales para Selección Dinámica (Transformer):**
   - El componente de **aprendizaje automático** introdujo la capacidad de *ajustar los parámetros de forma continua y en tiempo real*, infiriendo condiciones como el nivel de confusión o riesgo directamente del estado del problema.

- El uso de técnicas como la *Descomposición en Valores Singulares (SVD)* se demostró esencial para la escalabilidad, permitiendo manejar secuencias largas y datos de entrada variables sin agotar recursos computacionales.
- Las comparaciones mostraron que, si bien PSO puede bastar para ciertos rangos de incertidumbre, el *Transformer* con un conjunto de entrenamiento amplio y diverso puede superar la estrategia puramente metaheurística, al no requerir el conocimiento exacto de parámetros externos (por ejemplo, el valor de confusión).

4. **Combinación y Flexibilidad:**
   - Se evidenció la ventaja de **combinar varias funciones probabilísticas** bajo una *función maestra*, que decide cuál resulta más adecuada en cada situación. Esto incrementó la adaptabilidad frente a diferentes escenarios, desde riesgo bajo hasta situaciones de alta volatilidad.
   - En problemas de riesgo, tal flexibilidad permitió un *balance* entre maximizar la ganancia (funciones más agresivas) y minimizar la probabilidad de pérdidas (funciones conservadoras), evitando caer en un desempeño deficiente en entornos particularmente adversos.
5. **Aplicaciones y Futuras Líneas de Investigación:**
   - Los casos estudiados confirman la versatilidad de la propuesta, abriendo la puerta a nuevas aplicaciones, por ejemplo, en *planificación financiera*, *logística adaptativa* o *control de procesos*.
   - Entre las oportunidades de mejora destacan:
     - Profundizar en el entrenamiento del *Transformer* con **datasets más amplios** y *modelos de memoria* más complejos.
     - Hibridar PSO con otros métodos de optimización o aprendizaje (por ejemplo, algoritmos genéticos o recocido simulado) para cubrir un espectro todavía más amplio de condiciones.

En conclusión, este trabajo confirma que la *integración de modelos probabilísticos, metaheurísticas y técnicas de aprendizaje automático* es una vía poderosa para resolver problemas complejos de decisión, ya sea bajo incertidumbre, riesgo o restricciones cambiantes. La posibilidad de ajustar dinámicamente los parámetros y la capacidad de explorar soluciones en un espacio continuo permite al algoritmo adaptarse eficazmente a un amplio abanico de entornos, demostrando así su **robustez** y **versatilidad**.

5.1 Repositorio de Código

Para quienes deseen explorar la implementación práctica de los conceptos y algoritmos expuestos en este trabajo, se ha creado un repositorio en *GitHub* que contiene:

- El código fuente de las funciones y modelos desarrollados.
- Notebooks con las pruebas realizadas y ejemplos de uso.

El repositorio se encuentra disponible en la siguiente dirección:

`https://github.com/EnzoDtoste/Decision-Making-Optimization`

Referencias

[1] Rojas D'Toste E., Marrero Severo A. *Sinergia de Probabilidades, Optimización y Aprendizaje Automático para la Toma de Decisiones: Propuesta Algorítmica.* Ciencias Matemáticas, X(X), X–XX. Recuperado a partir de `https://revistas.uh.cu/rcm/`

[2] K. Sörensen, "Metaheuristics—The metaphor exposed," *International Transactions in Operational Research* **22**(1) (2015), 3–18. `https://doi.org/10.1111/itor.12001`

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," *arXiv preprint*, arXiv:1706.03762, 2017, https://arxiv.org/pdf/1706.03762

[4] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika* **1** (1936), 211–218. https://doi.org/10.1007/BF02288367

[5] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, Perth, Australia, 1995, pp. 1942–1948, https://www.cs.tufts.edu/comp/150GA/homeworks/hw3/_reading6%201995%20particle%20swarming.pdf