Artículo Original / Original Research

Sinergia de Probabilidades, Optimización y Aprendizaje Automático para la Toma de Decisiones: Propuesta Algorítmica

Synergy of Probabilities, Optimization, and Machine Learning for Decision Making: Algorithmic Proposal

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

Resumen

Este trabajo aborda el área de problemas de toma de decisiones en los que, en cada paso, un agente debe escoger una opción dentro de un conjunto finito—o incluso infinito numerable—de candidatos. Dado que las variantes exactas son NP-Completas, los métodos exhaustivos se vuelven inviables a gran escala. Se propone, por tanto, un *marco de asignación probabilística*: una función continua $f : [0, 1) \to [0, n)$ transforma un número aleatorio uniforme en un índice, asignando a cada decisión una probabilidad ajustable.

Los parámetros de f se optimizan con *Particle Swarm Optimisation* (PSO). Cuando los parámetros estáticos no bastan, un Transformer—cuya entrada se reduce mediante *SVD*—genera parámetros dinámicos en función del estado actual.

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

Abstract

This work tackles the area of decision-making problems in which an agent must choose, at each step, one option from a finite—or even countably infinite—set of candidates. Because the underlying optimisation tasks are NP-Complete, exact methods become infeasible as the problem size grows. We therefore propose a *probabilistic assignment framework*: a continuous function $f : [0, 1) \to [0, n)$ maps a uniform random value to a decision index, so that each option receives a tunable probability of being selected.

Parameters of f are optimised with Particle Swarm Optimisation (PSO). When static parameters are insufficient, a Transformer—fed with a sequence reduced by Singular Value Decomposition (SVD)—produces *dynamic* parameters conditioned on the current state.

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: Propuesta Algorítmica.* Ciencias Matemáticas, X(X), X–XX. Recuperado a partir de `https://revistas.uh.cu/rcm/`

1 Introducción

1.1 Contexto

Los problemas de toma de decisiones son fundamentales en diversas áreas de la ciencia, la ingeniería y la industria. Estos problemas surgen cuando es necesario elegir entre múltiples alternativas, optimizando un conjunto de objetivos bajo ciertas restricciones. Ejemplos notables incluyen la planificación de rutas en logística, la selección de inversiones en finanzas, el diagnóstico asistido por computadora en medicina, y la navegación autónoma en robótica.

En la actualidad, la resolución de problemas de toma de decisiones enfrenta diversos desafíos. La incertidumbre inherente a los datos y las condiciones dinámicas de los entornos reales complican el proceso de toma de decisiones, y la creciente complejidad de los problemas plantea altas demandas computacionales, lo que exige el desarrollo de algoritmos eficientes y escalables.

En este contexto, los enfoques basados en probabilidades, algoritmos de optimización y aprendizaje automático han demostrado ser herramientas poderosas. Las probabilidades permiten modelar la incertidumbre de manera formal, los algoritmos de optimización ofrecen métodos efectivos para encontrar soluciones óptimas, y el aprendizaje automático proporciona la capacidad de adaptar las estrategias a partir de datos. Sin embargo, integrar estas herramientas en un marco único y eficiente sigue siendo un desafío abierto.

Este trabajo busca abordar este desafío mediante el diseño de un algoritmo híbrido que combine probabilidades, optimización y aprendizaje automático para resolver problemas complejos de toma de decisiones. Esta combinación no solo tiene el potencial de superar las limitaciones de enfoques tradicionales, sino también de abrir nuevas posibilidades en aplicaciones reales, desde la industria hasta la investigación científica.

1.2 Planteamiento del Problema

El problema central abordado es el diseño y la estructuración de un algoritmo capaz de tomar decisiones óptimas dentro de un conjunto (finito o infinito contable) de opciones. Este algoritmo debe garantizar un resultado final que maximice el cumplimiento de los objetivos planteados en un problema específico.

La complejidad principal radica en la diversidad de los problemas y en la naturaleza dinámica del conjunto de decisiones. En muchos casos, este conjunto no es estático, sino que puede variar tanto en la cantidad de elementos como en sus características a medida que el problema evoluciona.

El objetivo es que el algoritmo sea lo suficientemente robusto y adaptable para encontrar estrategias efectivas independientemente de la naturaleza y las variaciones del problema. Esto requiere integrar de manera eficiente herramientas probabilísticas, de optimización y de aprendizaje automático, creando un enfoque capaz de abordar esta dinámica de manera generalizada y efectiva.

1.3 Propuesta y Enfoque

El algoritmo propuesto utiliza una función matemática adaptable que varía en función del problema a resolver. Esta función recibe como entrada un valor aleatorio generado de manera uniforme y devuelve un número entre 0 y n (sin incluir n), donde n representa el tamaño (finito o infinito contable) del conjunto de decisiones posibles. El valor devuelto se trunca a un entero, que se utiliza como índice para seleccionar una decisión de las opciones disponibles.

El orden de las decisiones está determinado por una función de aptitud, la cual evalúa qué tan buena es cada decisión basada en un criterio heurístico específico. La función matemática, por lo tanto, define intervalos probabilísticos de confianza para cada decisión. Sin embargo, debido a la complejidad inherente de los problemas abordados, seleccionar la decisión con la mejor aptitud no siempre garantiza un resultado óptimo, o incluso satisfactorio.

La optimización en este contexto consiste en ajustar los parámetros de la función matemática para definir los intervalos probabilísticos que maximicen los resultados. Este proceso otorga un poder significativo al criterio heurístico utilizado, el cual en muchos casos será lo suficientemente potente, como deberá ser demostrado. Sin embargo, esta estrategia presenta limitaciones, ya que al elegir los parámetros de la función de forma discreta, se

adopta un comportamiento probabilístico que no considera el estado actual o pasado del problema. Esto implica que dos instancias del mismo problema que podrían seguir trayectorias distintas usarán la misma estrategia, lo que disminuye la capacidad del algoritmo para adaptarse dinámicamente.

Para superar esta limitación, se introduce el uso del aprendizaje automático, específicamente redes neuronales, como parte del enfoque. Estas redes permiten ajustar los parámetros de la función de forma continua, basándose en el estado actual del problema. Esto proporciona al algoritmo una mayor flexibilidad y potencia, ya que los parámetros ya no se seleccionan de forma discreta, sino que se determinan mediante una "función continua"generada por la red neuronal. Este enfoque mejora la adaptabilidad del algoritmo y amplía significativamente su capacidad para abordar problemas complejos y dinámicos.

2 Función de Asignación Probabilística de Decisiones

En esta sección se describe una función matemática que, a partir de un valor aleatorio uniforme en el intervalo $[0, 1)$, permite asignar un índice válido dentro de un conjunto finito (o potencialmente infinito) de decisiones. El objetivo es que cada decisión reciba una probabilidad de ser elegida, cumpliendo con la condición de que la suma de todas estas probabilidades sea igual a 1.

2.1 Descripción General

Sea n el número de decisiones posibles, donde n puede ser un valor entero finito o tender a $+\infty$. Definimos una función

$$f : [0, 1) \to [0, n),$$

cuyo propósito es mapear un valor aleatorio $u \in [0, 1)$ (generado de manera uniforme) a un valor real en el rango $[0, n)$. A continuación, se trunca el valor obtenido, esto es,

$$\text{decision\_index} = \lfloor f(u) \rfloor,$$

para seleccionar un índice de decisión dentro del conjunto $\{0, 1, 2, \ldots, n-1\}$ (o hasta $+\infty - 1$ si n no está acotado superiormente).

2.2 Condiciones de la Función

Para garantizar una distribución probabilística coherente, la función $f(u)$ debe ser continua y cumplir con los siguientes requisitos:

1. **Dominio y Rango:** El dominio debe ser el intervalo $[0, 1)$, mientras que el rango se extiende hasta $[0, n)$. Esto asegura que un valor de entrada u (aleatorio uniforme) se convierta en un índice válido de decisión.

2. **Condiciones de Borde:**

$$u \in [0, 1), \quad \begin{cases} \exists\, u: & f(u) = 0, \\ \exists\, u: & \lim_{x \to u} f(x) = n \end{cases}$$

   Esta característica garantiza que la función cubra todas las decisiones posibles cuando u varía desde 0 hasta valores cercanos a 1.

3. **Asignación de Probabilidades:** El uso de un valor aleatorio uniforme u y la partición del intervalo $[0, 1)$ en subintervalos que correspondan a cada decisión asegura que la suma de las probabilidades asignadas a todas las decisiones sea 1.

2.3 Interpretación Probabilística

Si se analiza la función $f(u)$ como un mecanismo de asignación de probabilidades, cada decisión $d \in \{0, 1, \ldots, n-1\}$ queda representada por el subconjunto de valores $u \in [0, 1)$ tales que

$$\lfloor f(u) \rfloor = d.$$

La medición de Lebesgue [1] (básicamente, la longitud) de ese subconjunto en $[0, 1)$ representa la probabilidad de que la decisión d sea elegida. Por definición de la variable u como uniforme en $[0, 1)$, la suma de dichas mediciones para todas las decisiones será igual a la longitud total del intervalo, es decir, 1.

En consecuencia, el algoritmo puede controlar la probabilidad asignada a cada decisión ajustando la forma de la función f. Por ejemplo, si para ciertos valores de u se hace crecer más rápido (o más lento) la función, la distribución de probabilidades variará, otorgando mayor (o menor) peso a decisiones específicas.

2.4 Ventajas de la Aproximación

- **Flexibilidad:** Cualquier función $f(u)$ que cumpla con las condiciones anteriores es válida, lo cual permite diseñar distintos perfiles de asignación de probabilidad según el problema de decisión.
- **Escalabilidad:** Si n tiende a infinito (por ejemplo, en escenarios con un número muy grande o dinámico de decisiones), la forma de f puede adaptarse para seguir cumpliendo con las restricciones de borde.
- **Simplicidad:** El mapeo directo de un número real en $[0, 1)$ a un índice entero es intuitivo y fácil de implementar.

2.5 Inversión de la Función Logarítmica

Se presenta un ejemplo concreto de cómo construir la función $f(x)$ descrita en la sección empleando un logaritmo. El objetivo principal es ilustrar cómo asignar probabilidades a un conjunto de n decisiones (ordenadas según un criterio *fitness*) de tal manera que las primeras decisiones tengan mayor probabilidad de ser escogidas, sin descartar por completo las que se consideran menos prometedoras.

2.5.1 Motivación del Uso de un Logaritmo

La elección de una función logarítmica surge de la idea de favorecer de forma significativa las decisiones con mejor *fitness*, pero permitiendo que, con menor probabilidad, también se seleccionen opciones menos sobresalientes. De esta manera, se logra un equilibrio entre la *explotación* de las mejores decisiones y la *exploración* de soluciones potencialmente útiles en el largo plazo.

2.5.2 Definición General

Partimos de la forma:

$$x \;=\; b\,\log_a\Big(c\,(y - d_x)\Big) \;+\; d_y,$$

donde:

- $a > 0 \wedge a \neq 1$ es la base del logaritmo.
- $b \neq 0$ es un factor de escala que ajusta la pendiente de la curva.
- $c > 0$ es un factor de escala dentro del logaritmo.
- d_x y d_y son términos de traslación en los ejes de x y y, respectivamente.

El objetivo es *despejar* y en función de x. Además, se desea que esta función satisfaga las dos condiciones de borde anteriormente mencionadas:

$$\begin{cases} y(0) = 0, \\ \lim\limits_{x \to 1^-} y(x) = n, \end{cases}$$

2.5.3 Despeje de y

Iniciamos con la ecuación:

$$x \;=\; b\,\log_a\Big(c\,(y - d_x)\Big) \;+\; d_y$$

Reordenamos para aislar el logaritmo:

$$x - d_y \;=\; b\,\log_a\Big(c\,(y - d_x)\Big)$$

Aplicando la definición de logaritmo, pasamos a la forma exponencial:

$$c\,(y - d_x) \;=\; a^{\frac{x - d_y}{b}}, \quad \Longrightarrow \quad y \;=\; d_x \;+\; \frac{1}{c}\,a^{\frac{x - d_y}{b}}$$

De este modo, obtenemos y en función de x, con la salvedad de que $\big(d_x, d_y\big)$ deben elegirse (o ajustarse) para cumplir las condiciones de borde.

2.5.4 Condiciones de Borde

Deseamos que sea monótona creciente, además de cumplir con las condiciones generales:

$$y(0) = 0 \quad \text{y} \quad y(1) = n$$

Por tanto,

$$\begin{cases} y(0) \;=\; d_x \;+\; \dfrac{1}{c}\,a^{\frac{0 - d_y}{b}} \;=\; 0, \\ y(1) \;=\; d_x \;+\; \dfrac{1}{c}\,a^{\frac{1 - d_y}{b}} \;=\; n \end{cases}$$

- Primera Ecuación: $y(0) = 0$

$$d_x \;+\; \frac{1}{c}\,a^{\frac{-d_y}{b}} \;=\; 0 \quad \Longrightarrow \quad d_x \;=\; -\,\frac{1}{c}\,a^{\frac{-d_y}{b}} \tag{1}$$

- Segunda Ecuación: $y(1) = n$

$$n \;=\; d_x + \frac{1}{c}\, a^{\frac{1-d_y}{b}} \;=\; \Big(-\frac{1}{c}\, a^{\frac{-d_y}{b}}\Big) + \frac{1}{c}\, a^{\frac{1-d_y}{b}}$$

$$n \;=\; \frac{1}{c}\Big[a^{\frac{1-d_y}{b}} \;-\; a^{\frac{-d_y}{b}}\Big]$$

Agrupando términos:

$$\frac{1}{c}\, a^{\frac{-d_y}{b}}\Big[a^{\frac{1}{b}}-1\Big] \;=\; n, \quad\Longrightarrow\quad a^{\frac{-d_y}{b}} \;=\; \frac{c\,n}{a^{\frac{1}{b}}-1} \tag{2}$$

2.5.5 Solución para d_x y d_y

A partir de (2), resolvemos d_y:

$$\frac{-d_y}{b} = \log_a\Big(\frac{c\,n}{a^{\frac{1}{b}}-1}\Big), \quad\Longrightarrow\quad d_y = -b\,\log_a\Big(\frac{c\,n}{a^{\frac{1}{b}}-1}\Big)$$

Con este valor de d_y, usamos (1) para obtener d_x:

$$d_x \;=\; -\frac{1}{c}\, a^{\frac{-d_y}{b}} \;=\; -\frac{1}{c}\,\frac{c\,n}{a^{\frac{1}{b}}-1} \;=\; -\frac{n}{a^{\frac{1}{b}}-1}$$

Por tanto, las constantes quedan fijadas como:

$$d_y = -b\,\log_a\Big(\frac{c\,n}{a^{\frac{1}{b}}-1}\Big), \quad d_x = -\frac{n}{a^{\frac{1}{b}}-1}$$

2.5.6 Función Resultante $y(x)$

Sustituyendo d_x y d_y en la expresión

$$y \;=\; d_x \;+\; \frac{1}{c}\, a^{\frac{x-d_y}{b}},$$

obtenemos:

$$y(x) = -\frac{n}{a^{\frac{1}{b}}-1} \;+\; \frac{1}{c}\, a^{\frac{x \,+\, b\,\log_a\left(\frac{c\,n}{a^{\frac{1}{b}}-1}\right)}{b}}$$

Podemos simplificar la potencia:

$$a^{\frac{b\,\log_a\left(\frac{c\,n}{a^{\frac{1}{b}}-1}\right)}{b}} = a^{\log_a\left(\frac{c\,n}{a^{\frac{1}{b}}-1}\right)} = \frac{c\,n}{a^{\frac{1}{b}}-1}$$

Así, la forma final resulta:

$$y(x) = -\frac{n}{a^{\frac{1}{b}}-1} \;+\; \frac{1}{c}\,\Big(\frac{c\,n}{a^{\frac{1}{b}}-1}\Big)\, a^{\frac{x}{b}} \;=\; n\,\frac{a^{\frac{x}{b}} \;-\; 1}{a^{\frac{1}{b}}-1}$$

2.5.7 Interpretación

- **Dominio:** Para $x \in [0,1)$, $y(x)$ se desplaza desde 0 hasta valores cercanos a n. Esto modela una curva logarítmica (o su inversa exponencial) que puede adaptarse vía a y b.
- **Escalabilidad:** El parámetro n define el "tope" al que llega $y(x)$ cuando $x \to 1^-$, y los parámetros $\big(a, b\big)$ controlan la forma de la curva.
- **Eficiencia de Cálculo:** La ventaja de esta inversión es que se sustituyen operaciones logarítmicas por potencias, generalmente más eficientes en implementaciones de gran escala o en sistemas embebidos.

2.6 Caso con $n \to +\infty$: Función de Proporcionalidad Inversa

Cuando se desea modelar un conjunto de decisiones de tamaño indefinido (o incluso infinito), las expresiones presentadas pueden resultar poco prácticas, pues al hacer $\lim_{n\to+\infty}$ a menudo el valor de la función también tiende a infinito, imposibilitando la selección de un *índice finito* de decisión.

Para afrontar este escenario, se propone un enfoque alternativo: usar una **función de proporcionalidad inversa**. La idea central es que, a medida que el parámetro de entrada ($x \in [0,1)$ generado aleatoriamente) se acerca a 1, la función crezca de manera no acotada, simulando la disponibilidad de decisiones más allá de cualquier número finito. A su vez, cerca de $x = 0$, la función valdrá 0 (o un valor muy pequeño), representando la primera decisión.

2.6.1 Definición General

Se adopta la siguiente forma general para la función:

$$y(x) \;=\; \frac{a}{b\,\big(x - d_x\big)} \;+\; d_y,$$

donde:

- $x \in [0,1)$ es el parámetro de entrada.

- $a > 0$ y $b > 0$ son factores de escala que determinan la curvatura y la velocidad de crecimiento de la función.
- d_x y d_y son términos de traslación que ajustan el dominio y el rango de la función.

El objetivo es imponer condiciones de borde que permitan que:

$$y(0) \;=\; 0 \quad \text{y} \quad \lim_{x\to 1^-} y(x) \;=\; +\infty$$

2.6.2 Condiciones de Borde

- **Condición 1:** $y(0) = 0$

$$0 \;=\; \frac{a}{b\,(0 - d_x)} \;+\; d_y$$

$$\Longrightarrow \quad d_y \;=\; -\,\frac{a}{b\,(-\,d_x)} \;=\; \frac{a}{b\,d_x}$$

- **Condición 2:** $\lim_{x\to 1^-} y(x) = +\infty$

Para que la fracción $\frac{a}{b\,(x - d_x)}$ crezca sin acotación conforme $x \to 1^-$, se requiere que el denominador tienda a 0 desde el *lado positivo*:

$$b\,(1 - d_x) \quad \overset{x\to 1^-}{\longrightarrow} \quad 0^+,$$

lo que implica

$$1 - d_x = 0 \quad \Longrightarrow \quad d_x = 1$$

2.6.3 Cálculo de d_x y d_y

Combinar ambas condiciones permite determinar $(d_x,\, d_y)$.

De la segunda condición:

$$d_x \;=\; 1$$

De la primera condición:

$$0 \;=\; \frac{a}{b\,(0 - 1)} \;+\; d_y \;=\; -\,\frac{a}{b} \;+\; d_y$$

$$\Longrightarrow \quad d_y = \frac{a}{b}$$

2.6.4 Función Final $y(x)$

Sustituyendo $d_x = 1$ y $d_y = \frac{a}{b}$ en la expresión original, obtenemos:

$$y(x) \;=\; \frac{a}{b\,(x - 1)} \;+\; \frac{a}{b}$$

Sin embargo, observemos el signo: para $0 \le x < 1$, $x - 1 < 0$, lo que hace que el término $\frac{a}{b(x-1)}$ sea negativo y, al acercarse $x \to 1^-$, tienda a $-\infty$. Esto *no* cumple el requisito de crecer a $+\infty$, sino que diverge en el sentido opuesto.

2.6.5 Ajuste de la Estructura

Para que el límite sea $+\infty$ cuando $x \to 1^-$, conviene invertir la posición de $(1 - x)$ en el denominador:

$$y(x) \;=\; \frac{a}{b\,(1 - x)} \;+\; d_y$$

De este modo, al acercarse $x \to 1^-$, $(1 - x) \to 0^+$ y la fracción $\frac{a}{b\,(1-x)}$ crece hacia $+\infty$.

Ahora se imponen las mismas condiciones de borde:

$$\begin{cases} \text{(i)}\ y(0) = 0, \\ \text{(ii)}\ \lim\limits_{x\to 1^-} y(x) = +\infty \end{cases}$$

Condición (i): $y(0) = 0$

$$0 \;=\; \frac{a}{b\,(1 - 0)} + d_y \;=\; \frac{a}{b} + d_y \quad \Longrightarrow \quad d_y = -\,\frac{a}{b}$$

Condición (ii): $\lim_{x\to 1^-} y(x) = +\infty$ Con $(1 - x) \to 0^+$ en el denominador,

$$y(x) \;=\; \frac{a}{b\,(1 - x)} \;-\; \frac{a}{b} \quad \overset{x\to 1^-}{\longrightarrow} \quad +\infty,$$

lo cual cumple el requisito.

2.6.6 Forma Explícita

La función queda:

$$y(x) \;=\; \frac{a}{b\,(1 - x)} - \frac{a}{b} \;=\; \frac{a - a\,(1 - x)}{b\,(1 - x)} \;=\; \frac{a\,x}{b\,(1 - x)}$$

$$y(x) \;=\; \frac{a\,x}{b\,(1 - x)}$$

$$\text{para } x \in [0, 1), \quad \lim_{x\to 1^-} y(x) = +\infty, \quad y(0) = 0$$

2.6.7 Interpretación y Uso Práctico

- **Indice de Decisión Infinito:** Al no haber un tope en el valor de $y(x)$, conceptualmente puede considerarse que existen *ilimitadas* decisiones ordenadas, y la variable x determina cuál de ellas se escoge.

- **Distribución Sesgada:**
  - Para valores de x cercanos a 0, $y(x)$ será pequeña, favoreciendo la selección de las "primeras" decisiones.
  - A medida que x crece, la función aumenta más rápidamente, si bien la probabilidad de llegar a valores muy altos de y (i.e., de decisiones muy alejadas) existe, pero es pequeña (el usuario controla esta curva vía a y b).

- **Aplicaciones:** Este tipo de asignación es útil cuando se desea explorar un espacio teóricamente infinito de opciones, pero con alta preferencia por soluciones con *fitness* superior (asociadas a valores más bajos de y) y una reducida, aunque existente, probabilidad de explorar opciones lejanas en el orden.

En resumen, la **función de proporcionalidad inversa** ofrece una solución elegante para el caso de $n \to +\infty$, conservando la propiedad de asignar mayor prioridad a las primeras decisiones, pero sin truncar la posibilidad de explorar otras con un *índice* arbitrariamente grande.

2.7 Combinación de Múltiples Funciones Probabilísticas

En ciertas situaciones, distintas funciones pueden resultar óptimas según la condición o etapa específica del problema. Para aprovechar las ventajas de cada una, es posible crear una *combinación* que seleccione, en cada momento, cuál de ellas se empleará para determinar la probabilidad de escoger cada decisión. El proceso se basa en:

1. Definir un **conjunto de funciones** probabilísticas, por ejemplo, la *función logarítmica inversa* y otras (polinómicas, exponenciales, etc.).

2. Diseñar una **función maestra** (por ejemplo, otra función logarítmica inversa) que devuelva la *probabilidad* de seleccionar cada una de las funciones del conjunto.

3. Al momento de tomar una decisión, *elegir probabilísticamente* cuál función del conjunto usar, de acuerdo con la distribución que genera la función maestra.

4. Aplicar la función elegida para calcular la probabilidad final de cada decisión en el problema, asegurando que se cumpla la condición de que la suma de probabilidades de todas las decisiones sea 1.

De esta manera, la **combinación** de funciones se comporta *como si fuese una sola*, pero con un *potencial de adaptación superior*, puesto que en cada situación puede optar por la función que mejor se ajuste a la dinámica o la información disponible del entorno. Además, se mantienen las **reglas de consistencia probabilística**, porque tanto la función maestra como las funciones individuales se encargan de normalizar sus respectivos valores, cumpliendo así con los requerimientos de una distribución válida.

En síntesis, la combinación de múltiples funciones probabilísticas **maximiza la flexibilidad** y la capacidad de adecuación a problemas donde un solo tipo de función puede no ser suficiente para capturar todas las facetas de la decisión. Esta extensión amplía el alcance de la aproximación probabilística y permite un mejor desempeño en escenarios de mayor complejidad y variabilidad.

3 Optimización de Parámetros Mediante Metaheurísticas

En la sección anterior se describieron diversas funciones para asignar probabilidades a decisiones en un problema de optimización, cada una con un conjunto de parámetros (a, b, c, d_x, d_y, etc.) cuyo valor determina la forma específica de la curva de asignación probabilística. Sin embargo, cada problema de decisión presenta características particulares, por lo que la configuración de parámetros que maximice los resultados de un caso podría no ser óptima para otro. Con esto surge la necesidad de un **proceso de optimización** que encuentre,

de manera automática, aquellos valores de parámetros que produzcan los mejores resultados para el problema bajo estudio.

3.1 Necesidad de la Optimización de Parámetros

La búsqueda manual de parámetros adecuados puede resultar inviable cuando:

- El espacio de parámetros es muy grande (por ejemplo, varias dimensiones).
- Existe no linealidad en cómo los parámetros afectan el desempeño final.
- No se dispone de un modelo analítico simple para relacionar los parámetros con el valor objetivo a maximizar.

Por ende, se recurre a métodos de optimización que, dada una *función objetivo* y unas *restricciones*, exploran el espacio de soluciones en búsqueda de los parámetros que optimicen (maximicen o minimicen) el desempeño esperado.

3.2 Metaheurísticas: Una Vía de Uso General

Existen numerosos algoritmos de optimización, desde métodos deterministas hasta métodos estocásticos que no requieren información explícita de derivadas. *Las metaheurísticas* [2] conforman un conjunto de procedimientos de búsqueda de propósito general que pueden adaptarse a una amplia variedad de problemas y restricciones:

- **Idea fundamental:** una metaheurística no está diseñada para un problema específico, sino que, a través de reglas de exploración y explotación, es capaz de iterar sobre un espacio de soluciones, refinando paulatinamente la búsqueda.
- **Ventaja principal:** se pueden aplicar *incluso cuando la función objetivo sea una* caja negra (black-box), siempre que sea posible evaluar la calidad de una solución (por ejemplo, simulando o ejecutando el problema con los parámetros propuestos).
- **Convergencia:** muchas metaheurísticas cuentan con mecanismos que les permiten escapar de máximos locales y seguir explorando. Sin embargo, no siempre garantizan un óptimo global estricto, sino una *buena* solución en un tiempo razonable.

3.3 Enjambre de Partículas (PSO): Principios Básicos

La optimización por enjambre de partículas (*Particle Swarm Optimization*, PSO) [3] es particularmente útil cuando:

1. El espacio de parámetros es de dimensión moderada o alta.
2. No se cuenta con información sobre el gradiente de la función objetivo.
3. Se puede evaluar rápidamente la calidad (o *fitness*) de una configuración de parámetros.

Estructura de PSO.

- Una **partícula** representa un conjunto de parámetros $\vec{p}$ que definen la función de asignación de probabilidades.
- Cada partícula tiene ademas una **velocidad** $\vec{v}$ en el espacio de búsqueda.
- **Cálculo de fitness:** para cada $\vec{p}$, se evalúa el resultado final del problema de decisión usando los parámetros propuestos. Ese valor sirve de guía para mejorar la búsqueda.
- **Movimiento:** en cada iteración, las partículas ajustan su velocidad y posición basadas en:
  1. Su mejor solución personal (memoria individual).
  2. La mejor solución global encontrada por cualquier partícula del enjambre (memoria colectiva).

Ventajas de PSO.

- **Facilidad de implementación:** PSO se resume en pocas líneas de código y no requiere gradiente.

- **Convergencia rápida:** particularmente en espacios de dimensión moderada.
- **Escapatoria de máximos locales:** no está completamente garantizada, pero el componente estocástico y la influencia colectiva suelen permitir escapar de algunos valles locales.

3.4 Aplicación al Caso de Asignación Probabilística

Para optimizar la función de asignación probabilística mediante metaheurísticas como PSO, se definen los siguientes elementos:

1. **Parámetros a Optimizar:** $\{a, b, c, d_x, d_y, \dots\}$ (dependiendo de la función escogida).
2. **Función Objetivo:** generalmente, la medida de *efectividad* de la asignación de probabilidades en la resolución del problema de decisión. Por ejemplo, la ganancia promedio, la proporción de veces que se encuentra una solución factible de cierto nivel, etc.
3. **Restricciones:** valores válidos de los parámetros, relaciones entre ellos (por ejemplo, $a > 1$, $b > 0$), y aspectos prácticos (tiempo de cómputo).
4. **Proceso Iterativo:**
   a) Generar o actualizar un conjunto de candidatos (N partículas en PSO).
   b) Evaluar cada candidato en la función objetivo.
   c) Modificar las soluciones según las reglas de la metaheurística (velocidades en PSO).
   d) Repetir hasta cumplir un criterio de parada (número máximo de iteraciones, convergencia, etc.).

La elección de una metaheurística depende de la *naturaleza del problema*, de las restricciones computacionales y de la facilidad de implementación. PSO, dada su sencillez y eficacia en espacios continuos, es una de las alternativas más llamativas para hallar los parámetros óptimos de las funciones de asignación probabilística propuestas.

4 Selección Dinámica de Parámetros Mediante Redes Neuronales

Una limitación de determinar los parámetros de la función de asignación probabilística utilizando metaheurísticas, lo cual conduce a un conjunto discreto de valores "óptimos" para un problema dado, es que los parámetros suelen fijarse de forma estática. A medida que el problema evoluciona o se presentan diversas *subinstancias* dentro de un mismo escenario, podría resultar beneficioso *ajustar los parámetros en tiempo real* conforme cambia el estado del problema. En este apartado se explora la posibilidad de dotar al algoritmo de una **capacidad de selección dinámica de parámetros** a través de una *función entrenable*, que se aproximará mediante redes neuronales [4].

4.1 Motivación de la Selección Dinámica

La búsqueda de parámetros que se ajusten *globalmente* a todo un problema de decisión puede ser insuficiente cuando:

- El **entorno** o las **restricciones** varían a lo largo de la resolución del problema.
- Se deseen explotar *estados intermedios*: en cada etapa, la configuración "óptima" de parámetros para la función de decisión puede diferir de la que se tendría en etapas iniciales o finales.
- Exista un **alto costo computacional** al recomputar parámetros con metaheurísticas en cada paso (dada la complejidad de los problemas reales), de forma que se requiera un modelo entrenado que prediga rápidamente los parámetros oportunos.

4.2 Uso de Redes Neuronales para la Selección Dinámica

Para aproximar una función $\mathcal{F}$ que, dado el *estado actual* del problema, devuelva un vector de parámetros $\vec{\theta}$ ($a, b, c, \dots$ o equivalentes), se recurre a *redes neuronales*. Estas redes disponen de la capacidad de capturar relaciones complejas entre

la entrada (estado) y la salida (parámetros) tras un proceso de entrenamiento supervisado.

4.2.1 Entrenamiento Supervisado

El entrenamiento de la red neuronal requiere un **conjunto de datos** con pares (entrada, salida deseada). En este caso:

1. **Entrada**: Una representación del estado (o el histórico de estados) del problema en un instante dado.
2. **Salida**: El vector de parámetros $\vec{\theta}$ que optimiza la decisión en ese estado.

Sin embargo, **la pregunta clave es cómo se genera dicho conjunto de entrenamiento**. Una estrategia viable es:

- Para diversas *instancias* del problema, aplicar el enfoque discreto basado en metaheurísticas para encontrar los parámetros óptimos en cada *subetapa* del problema.
- Almacenar cada par (estado, parámetros óptimos) como un ejemplo de entrenamiento.
- Repetir en un número suficientemente grande de instancias (y subinstancias) para cubrir la variedad de situaciones posibles.

De este modo, la red *aprende* a reproducir la solución que la metaheurística habría producido, pero *en tiempo de ejecución* y de manera *continua*.

4.3 Dimensión Variable en la Entrada

Una de las dificultades más notables es que los **problemas pueden tener estados con distinto tamaño de información**:

- **Secuencias de longitud variable**: algunos problemas tienen un número de pasos que cambia dinámicamente.
- **Diferente representación de datos**: según la instancia, podrían variar la cantidad de entidades, restricciones activas, etc.

La mayoría de *redes neuronales clásicas* (ejemplo, perceptrones multicapa) tienen un tamaño fijo de entrada. Para abordar entradas de longitud variable se contemplan arquitecturas que puedan procesar secuencias de manera flexible:

- **Redes Recurrentes [5] (RNN, LSTM, GRU):** Adecuadas para procesamiento secuencial, donde la entrada se "alimenta" paso a paso. Son una opción para tratar datos con distintas longitudes, aunque pueden presentar problemas de gradiente en secuencias muy largas [6].
- **Redes Convolucionales [7] con mecanismos de *pooling* adaptativo**: Permiten analizar subsecuencias y extraer características, aunque se adecúan mejor a datos 2D (imágenes) o secuencias con cierto alineamiento.
- **Transformers**: Arquitectura que maneja secuencias usando *atención*, capaz de procesar representaciones de diverso tamaño y capturar dependencias a larga distancia de manera más efectiva.

4.4 Arquitectura de tipo Transformer

Los **Transformers** [8] han ganado popularidad por sus impresionantes resultados en PLN (Procesamiento de Lenguaje Natural), pero su uso se ha generalizado a otros dominios.

Aspectos clave:

- **Entrada como secuencia:** Se codifica el *input* en un conjunto de *tokens*. Cada *token* recibe una representación *embedding* que se enriquece con información posicional o de segmentación.
- **Atención multi-cabeza (Multi-head Attention):** Permite que cada posición (*token*) preste atención a cualquier otra posición en la secuencia, aprendiendo las relaciones importantes entre ellos.
- **Aplicación más allá del Lenguaje:** Si se aplanan los datos del estado del problema a una secuencia (*token* tras *token*), el Transformer puede procesarlo y mapearlo a

una salida de tamaño fijo (los parámetros a predecir).

4.4.1 Manejo de Estados Variados

Para lidiar con **información heterogénea** en un mismo problema, una estrategia:

- **Aplanar cada estado** (o cada componente relevante) en un vector que se convierte en un token.
- Insertar un *token separador* (ejemplo, un símbolo especial $\langle SEP \rangle$) entre estados sucesivos.
- Construir una *secuencia n-única* compuesta por todos los tokens de los estados relevantes, de modo que el Transformer "vea" el conjunto total de información como un bloque continuo.

La atención del Transformer permite que el modelo descubra las relaciones importantes entre los elementos de esta secuencia, sin requerir una arquitectura específica para la variable longitud.

4.5 Proceso General de Entrenamiento

1. **Definir la estructura de entrada**: representar cada estado como un vector de características. Encadenar todos los estados transcurridos con un token separador.
2. **Diseñar la red Transformer**:
   - Bloques de atención multi-cabeza.
   - Capas feed-forward intermedias.
   - Mecanismos de *positional encoding* para indicar el orden o la secuencia temporal.
3. **Salida de la red**: un bloque feed-forward final (o cabeza de salida) produce un vector de parámetros $\vec{\theta}$.
4. **Objetivo de aprendizaje**: Mínimo error (por ejemplo, en norma cuadrática) entre la salida de la red y los parámetros $\vec{\theta}^*$ que determinan la solución óptima en cada subinstancia, obtenida por la metaheurística.
5. **Entrenamiento**:
   - Se divide el conjunto de datos (instancias + subinstancias) en entrenamiento, validación y test.
   - Se aplica un método de optimización estándar (por ejemplo, Adam) para ajustar los pesos de la red.
   - Al final, se evalúa la calidad prediciendo los parámetros en instancias nunca antes vistas y midiendo el desempeño del problema de decisión.

4.6 Reflexiones y Proyecciones

- **Flexibilidad:** El uso de un Transformer no se limita a problemas con una secuencia temporal; cualquier información que pueda representarse como tokens puede beneficiarse de la atención multi-cabeza.
- **Escalabilidad:** Aunque los Transformers han probado su eficacia a gran escala, pueden volverse costosos en términos de memoria y cómputo si la secuencia es muy larga. Se han propuesto variantes (Longformer, Performer, etc.) para mitigar estas limitaciones.
- **Calidad del conjunto de entrenamiento:** Cuanta más instancias y subinstancias se analicen con metaheurísticas de alta calidad, mejor podrá la red neuronal aproximar la selección de parámetros en tiempo real.

En conclusión, la selección dinámica de parámetros vía redes neuronales es un paso más *ambicioso* que busca convertir la asignación discreta (obtenida por metaheurísticas) en una selección continua, adaptable al estado presente. El uso de **Transformers** se presenta como una estrategia particularmente prometedora para manejar entradas de tamaño variable y capturar relaciones complejas entre estados, permitiendo que el modelo "aprenda" cómo deberían evolucionar los parámetros en el tiempo o conforme varíen las condiciones del problema.

4.7 Manejo de la Escalabilidad: Reducción de Dimensionalidad con SVD

Pese a su *gran potencial*, los Transformers presentan una limitación crucial en términos de **esca-**

bilidad: la capa de atención requiere construir y procesar una matriz de atención de tamaño $N \times N$ para una secuencia de longitud N. En problemas donde la secuencia de estados crece a lo largo del tiempo (o abarca gran cantidad de elementos), la memoria necesaria para representar y procesar dicha matriz puede volverse prohibitivamente grande.

4.7.1 Transformar la Secuencia de Estados en una Matriz 2D

Una forma de abordar este inconveniente es *transformar la secuencia de estados* en una matriz 2D que capture de forma compacta la información esencial. Por ejemplo, si en cada paso del problema se produce un vector de características (información sobre el entorno o las decisiones pasadas), podemos concatenar o apilar dichos vectores en la dirección de las filas o las columnas, obteniendo una representación del estado global:

$$\mathbf{M} \in \mathbb{R}^{N \times d},$$

donde N es la longitud de la secuencia (número de estados considerados) y d es la dimensión de cada vector de características.

4.7.2 Aplicación de la Descomposición en Valores Singulares (SVD)

Para reducir la dimensionalidad y *controlar* el tamaño de la representación, se emplea la **Descomposición en Valores Singulares (SVD, por sus siglas en inglés)** [9]:

$$\mathbf{M} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^{\top},$$

donde:

- $\mathbf{U} \in \mathbb{R}^{N \times N}$ es una matriz unitaria (columnas ortonormales).
- $\mathbf{\Sigma} \in \mathbb{R}^{N \times d}$ es una matriz diagonal (o cuasi-diagonal) con los valores singulares de $\mathbf{M}$.
- $\mathbf{V} \in \mathbb{R}^{d \times d}$ es también una matriz unitaria.

Los valores singulares se ordenan de mayor a menor. De esta forma, **los primeros valores singulares** capturan la mayor parte de la energía (información) de la matriz $\mathbf{M}$.

Selección de los vectores más relevantes Para reducir la dimensionalidad de $\mathbf{M}$, se toma un número $r < \text{mín}(N, d)$ de los valores singulares más altos. Esto equivale a truncar $\mathbf{\Sigma}$ y las columnas/filas correspondientes de $\mathbf{U}$ y $\mathbf{V}$, quedando:

$$\mathbf{M} \approx \mathbf{U}_r\,\mathbf{\Sigma}_r\,\mathbf{V}_r^{\top},$$

donde $\mathbf{U}_r \in \mathbb{R}^{N \times r}$, $\mathbf{\Sigma}_r \in \mathbb{R}^{r \times r}$ y $\mathbf{V}_r \in \mathbb{R}^{d \times r}$. Esta aproximación minimiza en sentido de *norma de Frobenius* [10] ($\|\mathbf{M} - \mathbf{M}_r\|_F$) el error al representar $\mathbf{M}$ con rango r.

4.7.3 Representación Fija y Compacta

Para propósitos de *entrada al Transformer* (o cualquier otra red neuronal de tamaño fijo), puede diseñarse un esquema que:

1. Elija un r constante para todas las instancias del problema.
2. Construya una representación vectorial tomando los vectores columna (o fila) en $\mathbf{U}_r$, $\mathbf{\Sigma}_r$, $\mathbf{V}_r$, o alguna combinación de ellos. Por ejemplo, concatenar las columnas principales de $\mathbf{U}_r$ y $\mathbf{V}_r$ para obtener una descripción de dimensiones fijas.
3. Ignore los valores singulares de menor magnitud (colocados al final de $\mathbf{\Sigma}$) que no aportan significativamente a la estructura del estado.

Con este procedimiento, **no importa cuán larga sea la secuencia original** (N), porque finalmente se *proyecta* a una representación $\mathbf{M}_r \in \mathbb{R}^{r \times r}$, o bien a un vector de dimensión fija (por ejemplo, $2\,r\,d$ si se concatenan parte de $\mathbf{U}_r$ y $\mathbf{V}_r$). Adicionalmente, los vectores ortonormales en SVD poseen propiedades algebraicas que pueden resultar más fáciles de explotar o aprender para la red neuronal que los datos originales "sin procesar".

4.7.4 Beneficios para el Modelo

- **Ahorro de Memoria:** Al representar la información relevante en un espacio de dimensión r, se evita la construcción de matrices $N \times N$ cuando N es grande.
- **Generalización Más Rápida:** La proyección SVD tiende a eliminar el ruido o la

redundancia en los datos, facilitando a la red neuronal detectar relaciones más claras y robustas.

- **Tamaño de Entrada Fijo:** Aun si el problema evoluciona indefinidamente y la secuencia de estados se vuelve muy larga, la representación de SVD puede mantener un tamaño constante, evitando la explosión de recursos.

4.7.5 Reflexiones Finales

La **aplicación de SVD** u otras técnicas de reducción de dimensionalidad (por ejemplo, PCA, proyecciones aleatorias, autoencoders) resulta clave para hacer *escalable* el uso de Transformers en problemas donde la secuencia de estados es muy grande. Aunque cada problema demandará su propio criterio de cuántos vectores (o qué umbral de energía) conservar en la descomposición, este enfoque proporciona un *control preciso* sobre la representación de la información, combinando:

1. La *capacidad explicativa* de la reducción SVD (que concentra la mayor parte de la variabilidad en menos dimensiones).

2. La *potencia* del mecanismo de atención de los Transformers, sin quedar limitado por los requerimientos explosivos de memoria en secuencias largas.

En conclusión, integrar la reducción de dimensionalidad mediante SVD antes de alimentar un Transformer constituye una solución elegante al problema de la *escalabilidad* en el contexto de selección dinámica de parámetros. Esta aproximación equilibra la retención de información clave con la necesidad práctica de mantener un tamaño de entrada manejable en la red neuronal.

5 Conclusiones Generales

El marco teórico establece, en primer lugar, la noción de una **función de asignación probabilística de decisiones** como mecanismo central para transformar una fuente uniforme de aleatoriedad en una distribución controlada sobre un conjunto ordenado de alternativas. Esta formulación abstrae la complejidad del problema específico y permite razonar de manera unificada sobre cómo concentrar o dispersar probabilidad según la necesidad de explotación (favorecer las decisiones mejor valoradas) o de exploración (mantener apertura a opciones menos evidentes). El valor conceptual clave es que la forma de la función, parametrizada adecuadamente, actúa como un regulador continuo del equilibrio entre ambos extremos sin requerir suposiciones rígidas sobre la estructura interna del problema.

En segundo lugar, se fundamenta el uso de **metaheurísticas** —en particular, PSO— como procedimiento genérico para ajustar los parámetros de dichas funciones cuando la relación entre la configuración paramétrica y el desempeño operativo es opaca, no lineal o difícil de derivar analíticamente. La optimización se plantea así como una caja negra en la que únicamente es necesario evaluar la calidad de la decisión inducida. PSO aporta un balance práctico entre simplicidad de implementación, exploración colectiva del espacio de parámetros y rapidez de convergencia en escenarios de dimensión moderada, sin exigir gradientes ni modelos internos diferenciables. Este acoplamiento (función probabilística + ajuste metaheurístico) dota al sistema de una fase de calibración inicial capaz de aproximar configuraciones robustas para un espectro amplio de instancias.

Finalmente, el marco extiende su alcance incorporando **selección dinámica de parámetros** mediante modelos de tipo Transformer, introduciendo adaptatividad contextual: los parámetros dejan de ser constantes globales y pasan a depender explícitamente del estado o historial observado. La arquitectura de atención permite modelar dependencias a distinta escala temporal sin imponer un orden secuencial rígido, pero su coste cuadrático motiva la integración de una etapa previa de **reducción de dimensionalidad con SVD**. Esta reducción condensa las secuencias históricas en representaciones compactas que preservan la información dominante y estabilizan la dimensión de entrada. El resultado es un ciclo coherente: la teoría define el espacio funcional; la optimización metaheurística proporciona una base paramétrica competente; y el módulo dinámico (Transformer + SVD) refina en tiempo real la asignación probabilística adaptándola a variaciones del entorno, reforzando la resiliencia del algoritmo frente a in-

certidumbre.

En conjunto, estos componentes articulan un marco flexible, escalable y conceptualmente transparente para la toma probabilística de decisiones: la función de asignación ofrece la semántica probabilística; PSO establece una calibración inicial eficaz sin requerir modelos diferenciables; y la selección dinámica basada en Transformers, apoyada en SVD, añade la capacidad de evolucionar los parámetros conforme cambian las condiciones, consolidando así un enfoque integral.

Referencias

[1] H. Lebesgue, «Intégrale, longueur, aire», *Annali di Matematica Pura ed Applicata (Serie 3)*, **7** (1902), 231–359. https://doi.org/10.1007/BF02420592

[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] 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

[4] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature* **323** (1986), 533–536. https://doi.org/10.1038/323533a0

[5] J. L. Elman, "Finding structure in time," *Cognitive Science* **14**(2) (1990), 179–211. https://doi.org/10.1207/s15516709cog1402_1

[6] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.* **5**(2) (1994), 157–166. https://doi.org/10.1109/72.279181

[7] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, Cambridge (MA), 2016. https://www.deeplearningbook.org

[8] 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

[9] 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

[10] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed., Cambridge Univ. Press, Cambridge, 2013. ISBN 978-0521548236. https://www.cambridge.org/highereducation/books/matrix-analysis/FDA3627DC2B9F5C3DF2FD8C3CC136B48