Escalabilidad de Redes NNUE para Búsqueda Informada en Puzles de Permutación N×N×N

Miguel Ángel De la Vega Rodríguez[2], Ignacio Aguilera-Martos[1,2], and Francisco Herrera[1,2]

[1] Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, Granada, España.
[2] Instituto Andaluz Interuniversitario en Ciencia de Datos e Inteligencia Computacional (DaSCI), Universidad de Granada, Granada, España.
miguevrod@correo.ugr.es, nacheteam@ugr.es, herrera@decsai.ugr.es

Resumen Las Redes Neuronales Eficientemente Actualizables (NNUE) han revolucionado la evaluación de posiciones en juegos de tablero discretos como el Ajedrez y el Shogi gracias a su capacidad para calcular actualizaciones incrementales en complejidad temporal $O(1)$. Sin embargo, su aplicabilidad en puzles de permutación de alta dimensionalidad, como el Cubo de Rubik $N \times N \times N$, no ha sido explorada. Este artículo analiza formalmente los límites matemáticos de las arquitecturas NNUE cuando se aplican a dominios con alta densidad de movimientos. Mientras que en un cubo estándar de $3 \times 3 \times 3$ un giro de cara altera aproximadamente el 34,6 % de las piezas de la corteza exterior, demostramos una ley de escala geométrica invariante, por la que a medida que la dimensión del puzle N crece, el ratio de características modificadas respecto al estado superficial total decrece cuadráticamente. Validado empíricamente mediante un enfoque de sistemas de alto rendimiento, los micro-benchmarks demuestran una ganancia en velocidad creciente con la dimensión del puzle, de hasta 83,4× sobre pasos densos tradicionales en capas ocultas masivas. Integrado en un solucionador híbrido que acopla la heurística regresora a un algoritmo de búsqueda informada y una base de datos perimetral, evaluamos su comportamiento sobre un dataset estandarizado de 800 instancias en el rango de mezclas $K \in [7, 14]$. Los hallazgos demuestran la viabilidad de las topologías ligeras para la extracción de patrones topológicos subyacentes y abren una nueva vía para la optimización de sistemas combinatorios en tiempo real.

Keywords: Redes Neuronales Eficientemente Actualizables · Cubo de Rubik · Puzles de Permutación · Combinatoria · Micro-benchmarking.

1. Introducción

En el ámbito de la Inteligencia Artificial aplicada a juegos con espacios de estados discretos, el compromiso entre la capacidad de representación de una función de evaluación y su coste computacional ha sido históricamente el principal factor limitante [1]. La introducción de las Redes Neuronales Eficientemente

Actualizables (NNUE), propuestas originalmente por Nasu para el juego del Shogi [2], supuso un cambio de paradigma en este sentido. Al sustituir el costoso paso de propagación hacia adelante (*forward pass*) denso por un transformador de características incremental, estas redes permiten que la CPU compute el estado intermedio recalculando exclusivamente los elementos que han cambiado entre dos posiciones sucesivas. Esta técnica reduce el coste de evaluación a una complejidad constante $O(1)$ en dominios como el ajedrez, donde cada jugada altera una fracción ínfima y dispersa del tablero, permitiendo tasas de exploración masivas en hilos convencionales de procesamiento [3].

A pesar de este éxito en entornos estáticos de carácter bidimensional, la traslación de la arquitectura NNUE a puzles de permutación tridimensionales y secuenciales, representados por la familia del Cubo de Rubik, presenta desafíos teóricos y prácticos complejos que han sido abordados de forma incompleta por la literatura tradicional [4]. A diferencia de un tablero de ajedrez, un movimiento básico en un Cubo de Rubik consistente en la rotación de una capa exterior desplaza un subconjunto denso de elementos de forma simultánea. Este hecho plantea la incógnita de si se mantiene la viabilidad de un operador incremental cuando el principio de dispersión que lo justifica de manera estructural se vería teóricamente degradado por la propia naturaleza geométrica del problema.

Este artículo analiza la viabilidad de las arquitecturas NNUE en puzles de permutación $N \times N \times N$ mediante una validación analítica y empírica de su escalabilidad. Para ello, se implementa un entorno de simulación que compara el coste de evaluación incremental frente a un enfoque denso clásico, y se entrena un regresor heurístico ligero como prueba de concepto. Dicho estimador se integra en un algoritmo de búsqueda informada con una base de datos perimetral precomputada a profundidad 6 mediante *Zobrist Hashing*, evaluándose sobre 800 instancias de test. Las principales contribuciones de este trabajo son la formulación de una ley de escala geométrica para caracterizar la densidad relativa de actualización conforme aumenta N, un estudio sistemático de rendimiento que muestra aceleraciones de hasta 83,4× frente a evaluaciones densas y la validación operativa de un solver híbrido orientado a estudiar el potencial de heurísticas incrementales ligeras en espacios combinatorios de alta dimensionalidad.

El resto del artículo se organiza de la siguiente manera. En la Sección 2 se revisa el marco teórico y el trabajo relacionado en materia de funciones incrementales y solucionadores tradicionales. En la Sección 3 se describe en detalle el diseño metodológico de la investigación. En la Sección 4 se exponen y discuten los resultados experimentales obtenidos. Por último, en la Sección 5 se resumen las conclusiones principales de este trabajo y se esbozan las líneas de investigación futuras.

2. Contextualización Teórica y Trabajo Relacionado

2.1. El Espacio de Estados en Puzles de Permutación $N \times N \times N$

La familia de puzles de permutación secuencial tridimensionales, parametrizada por el rango de sus dimensiones como $N \times N \times N$, presenta una elevada

complejidad asintótica en su espacio de búsqueda combinatorio bajo el análisis de la teoría de grupos [8]. Para el caso estándar del cubo de $3\times3\times3$, el espacio de configuraciones posibles viene determinado por la combinatoria de sus 8 esquinas, con 3 orientaciones posibles cada una, y sus 12 aristas, con 2 orientaciones posibles, deduciendo las restricciones de paridad geométrica que hacen inalcanzables ciertos estados mediante giros válidos: $|\Omega_{3\times3\times3}| = \frac{8!\cdot3^7\cdot12!\cdot2^{11}}{2} \approx 4{,}33 \times 10^{19}$.

A medida que el rango del hipercubo escala hacia dimensiones superiores ($N > 3$), la complejidad combinatoria sufre una explosión hiper-exponencial debido a la aparición de centros móviles y aristas múltiples desprovistas de un punto de anclaje estático interno.

En la métrica de giros métricos (*Half-Turn Metric* o HTM), un cubo estándar dispone de un factor de ramificación elemental de 18 movimientos posibles (6 caras $\times$ 3 tipos de giro). Tras aplicar reducciones básicas basadas en el historial de movimientos de la pila para eliminar redundancias inversas y conmutativas sobre la misma capa consecutiva, el factor de ramificación efectivo se estabiliza en aproximadamente 13,35.

El límite superior absoluto para la resolución óptima de cualquier instancia del cubo $3\times3\times3$ bajo la métrica HTM está acotado por el denominado *Número de Dios*, cuyo valor se demostró matemáticamente que es exactamente igual a 20 movimientos [9].

2.2. El Paradigma de Redes Neuronales Eficientemente Actualizables y su Operador

La dicotomía entre la velocidad de exploración y la precisión de la heurística ha fragmentado los motores de ajedrez modernos en tres vertientes arquitectónicas. Estas escuelas abarcan las Heurísticas Clásicas Artesanales con tasas superiores a 10^7 nodos por segundo (NPS) basadas en lógica cableada, las Redes Convolucionales Profundas o Transformers implementadas en motores como *Leela Chess Zero* [13] que capturan relaciones estratégicas abstractas con gran precisión pero reducen el caudal a una escala de entre 10^3 y 10^4 NPS requiriendo aceleración por hardware gráfico, y finalmente la Inferencia Incremental NNUE. Este último enfoque, adaptado en motores como Stockfish, ejecuta la actualización del acumulador directamente en los hilos de la CPU mediante instrucciones SIMD, operando en magnitudes de millones de NPS en hardware de consumo [3].

La arquitectura de una red NNUE rompe el flujo convencional de las redes perceptrón multicapa, MLP por sus siglas en inglés, mediante la separación y optimización de su primera capa de transformación. Consideremos un vector de entrada disperso y binario $\boldsymbol{x} \in \{0,1\}^D$, donde la dimensionalidad de las características del dominio se fija en $D = 480$. Este valor no constituye un parámetro arbitrario de la arquitectura, sino que responde de forma directa a la codificación matemática típica para la representación indexada de las orientaciones y posiciones de las 12 aristas y 8 esquinas del cubo sobre la estructura del acumulador lineal. En un instante t de la secuencia de evaluación, la activación de la primera capa oculta (el acumulador) se define clásicamente como $\boldsymbol{a}^{(t)} = W_1\boldsymbol{x}^{(t)} + \boldsymbol{b}_1$. Donde $W_1 \in \mathbb{R}^{M\times D}$ es la matriz de pesos y $\boldsymbol{b}_1 \in \mathbb{R}^M$ el vector de sesgos. En

lugar de computar el producto matricial denso, cuya complejidad es $O(M \cdot D)$, la NNUE aprovecha la transición secuencial entre estados. Si un movimiento altera un conjunto de índices activos, denotado por las características eliminadas ($\mathcal{C}_{\text{del}}$) y añadidas ($\mathcal{C}_{\text{add}}$), el estado del acumulador en el instante $t+1$ se evalúa de forma estrictamente incremental:

$$\boldsymbol{a}^{(t+1)} = \boldsymbol{a}^{(t)} - \sum_{i \in \mathcal{C}_{\text{del}}} W_1[:, i] + \sum_{j \in \mathcal{C}_{\text{add}}} W_1[:, j] \tag{1}$$

Esta formulación reduce la complejidad del paso de inferencia inicial a un coste proporcional a los elementos modificados, independientemente de la dimensionalidad total de la entrada D. Las capas subsiguientes procesan este acumulador mediante funciones de activación acotadas linealmente (*Clipped ReLU*) y capas densas de dimensiones reducidas.

2.3. Sistemas Heurísticos e Inteligencia Artificial en el Cubo de Rubik

La resolución automática del Cubo de Rubik ha estado dominada históricamente por enfoques analíticos puros como el algoritmo de dos fases de Kociemba [5] o búsquedas informadas tradicionales basadas en el algoritmo IDA^* expuesto por Korf [4], las cuales se apoyan de manera intensiva en bases de datos de patrones (*pattern databases*) que actúan como heurísticas admisibles fijas.

En el ámbito de la Inteligencia Artificial profunda, el hito principal lo constituye *DeepCubeA* [7], el cual destaca por su alta capacidad de generalización al resolver múltiples puzles combinatorios sin reconfigurar su núcleo de aprendizaje. Sin embargo, desde la perspectiva de la ingeniería de sistemas, este modelo adolece del mismo cuello de botella que las redes densas de evaluación de juegos, dado que la latencia de inferencia por nodo es masiva en entornos de ejecución de CPU (10–50 ms). Esta restricción fuerza al algoritmo de búsqueda a explorar un volumen sumamente acotado de estados. Por otro lado, los modelos contemporáneos basados en Transformers puros y mapeos directos de atención, englobados bajo el término de aproximaciones sin búsqueda o *searchless* [14], eligen la exploración del árbol por completo, pero carecen de garantías matemáticas o de convergencia de resolución ante estados que se desvíen del espacio de datos muestreado durante el entrenamiento. Esta brecha tecnológica justifica el estudio de la viabilidad de la arquitectura ágil NNUE para dotar a los solucionadores combinatorios de un rendimiento masivo.

3. Metodología y Diseño Algorítmico

El diseño experimental de esta investigación se estructura en tres componentes interconectados: un entorno de evaluación comparativa de hardware, una metodología de entrenamiento supervisado orientada al modelado de la función de coste heurística, y un algoritmo de búsqueda híbrido optimizado.

3.1. Diseño del Banco de Pruebas de Escalabilidad

Para analizar el impacto de la densidad geométrica, se ha construido un entorno de simulación capaz de aislar de forma estricta las dos arquitecturas de inferencia bajo idénticas restricciones de dimensiones espaciales (N) y de capacidad de red (M):

1. **Canalización de Evaluación Densa:** Modela una red perceptrón convencional en el estado del arte. Ante una dimensión N, expande por completo el estado visible de la corteza exterior ($D = 6N^2 - 12N + 8$) y ejecuta una multiplicación de matrices de precisión simple aplicando un bucle denso de operaciones de Multiplicación y Acumulación, MAC por sus siglas en inglés.
2. **Canalización de Evaluación Incremental (NNUE):** Prescinde de la exploración lineal. Modifica el acumulador local calculando el mapa de índices alterados por la rotación geométrica (N^2 elementos) y ejecuta operaciones aritméticas de adición y sustracción sobre los vectores de pesos.

De forma más precisa, sea $S(N) = 6N^2 - 12N + 8$ el número de elementos visibles de la corteza exterior de un cubo $N \times N \times N$, y sea $\Delta(N) = N^2$ el número de posiciones afectadas por un giro de capa exterior. La densidad relativa de actualización puede definirse como $\rho(N) = \frac{\Delta(N)}{S(N)} = \frac{N^2}{6N^2-12N+8}$. Cabe precisar que $\Delta(N)$ contabiliza posiciones cuyo estado visible se altera tras el giro; en el caso de la pieza central de una cara, esta reorienta sobre su propio eje sin cambiar de posición dentro de la corteza, por lo que su contribución a la densidad de actualización procede del cambio de orientación y no de un desplazamiento posicional. Aunque $\Delta(N)$ crece cuadráticamente con N, la actualización incremental evita recorrer la representación completa y restringe el cómputo al subconjunto modificado. Por ello, el banco de pruebas compara únicamente la canalización densa y la incremental NNUE sobre una misma CPU de consumo, con el fin de aislar el efecto del operador incremental frente al flujo denso tradicional. Se descartan deliberadamente GPU y arquitecturas neuronales más complejas para evitar factores externos como la latencia de transferencia o la sobrecarga de los frameworks de aprendizaje profundo [10].

Para reforzar la validez del benchmark, el motor elimina las asignaciones dinámicas de memoria en tiempo de ejecución y almacena las estructuras auxiliares y activaciones en arrays estáticos ubicados en la pila. El código fuente del entorno de ejecución se encuentra disponible públicamente para garantizar la reproducibilidad y trazabilidad de los experimentos[3].

3.2. Topología de la Red y Metodología de Aprendizaje

El objetivo de aprendizaje consiste en aproximar la función de regresión no lineal $H : \Omega \to \mathbb{R}$, encargada de estimar el número de movimientos restantes requeridos para alcanzar el estado objetivo. La topología de la red (*NnueRegressor*) se compone de una Capa de Entrada Dispersa ($D = 480$), una Capa de

[3] `https://github.com/Miguevrgo/Rubik`

Transformación Lineal / Acumulador ($M = 256$) procesada mediante *Clipped ReLU*, una capa oculta densa reducida de 128 neuronas con activación ReLU para sintetizar correlaciones de alto nivel, y una neurona de salida lineal que proporciona el escalar de distancia estimado.

El entrenamiento sigue una estrategia de mezcla inversa parametrizada (*Auto-Scrambler*) aplicando secuencias de $K \in [1, 20]$ movimientos aleatorios. El entrenamiento se optimiza mediante la función de pérdida del Error Cuadrático Medio (MSE) y un optimizador Adam dotado de parada temprana. La configuración completa para el entrenamiento puede consultarse en la Tabla 1.

Cuadro 1. Configuración del entrenamiento y evaluación del modelo heurístico.

Elemento	Configuración
Generación de datos	AutoScrambler con $K \in [1, 20]$
Etiqueta	Profundidad de mezcla K como proxy de dificultad
Entrada	Codificación dispersa $D = 480$
Modelo	NNUERegressor
Arquitectura	Acumulador 256, capa densa 128, salida lineal
Pérdida	Error cuadrático medio (MSE)
Optimizador	Descenso de gradiente estocástico
Early stopping	Sí
Test heurístico	1.000 instancias
Test solver	800 instancias, $K \in [7, 14]$

Cabe señalar que la profundidad de mezcla K se utiliza como una etiqueta proxy de dificultad y no debe interpretarse necesariamente como la distancia óptima al estado resuelto. Debido a cancelaciones, conmutatividades y redundancias entre movimientos, una secuencia aleatoria de longitud K puede generar estados resolubles en menos de K pasos. Por tanto, el regresor aprende una aproximación supervisada a la complejidad inducida por el proceso de mezcla, no una función exacta de distancia óptima.

3.3. Algoritmo de Búsqueda Híbrido y Poda por Historial

La resolución final de las instancias se delega en un algoritmo de búsqueda iterativa en profundidad informada (IDA^*), donde el coste de un camino viene dado por $F(n) = G(n) + H(n)$ [6]. Para atenuar la explosión combinatoria, se introduce una restricción estricta en el bucle de expansión basada en el historial de movimientos de la pila recursiva: se prohíbe la generación de movimientos inversos inmediatos sobre la misma cara y se rompe la simetría conmutativa en caras opuestas pariguales ($U/D, L/R, F/B$), reduciendo el factor de ramificación efectivo a 13,35. El término heurístico $H(n)$ se calcula como el máximo puntual entre dos bases de datos de patrones admisibles (esquinas y aristas) y la predicción del regresor NNUE escalada empíricamente por un factor de 0,7, esto es, $H(n) = \max(h_{\text{esquinas}}, h_{\text{aristas}}, 0{,}7 \cdot h_{\text{NNUE}})$, de modo que la base de datos exacta prevalece siempre que ofrezca una cota igual o superior a la heurística aprendida.

La búsqueda se acopla a una Base de Datos de Finales Perimetral precomputada a profundidad 6 desde el estado resuelto, que almacena los 983,926 estados

únicos del subgrafo de destino indexados mediante *Zobrist Hashing*. Al evaluar un nodo, el algoritmo realiza una consulta única: si el hash colisiona con la tabla hash de transposiciones de la RAM, se detiene la inferencia de la IA y se ejecuta la secuencia inversa determinista en $O(1)$.

4. Resultados Experimentales y Discusión

El marco experimental se ha dividido en tres bloques secuenciales: eficiencia computacional pura del hardware, métricas de convergencia de la red entrenada y evaluación del solucionador híbrido sobre conjuntos de datos estandarizados.

4.1. Benchmarking de Hardware e Invarianza del Speedup

Se evaluó el tiempo de cómputo requerido por la CPU variando el rango dimensional del cubo ($N \in [3, 12]$) y la capacidad del acumulador oculto ($M \in [512, 8192]$). Los resultados numéricos promediados tras la ejecución masiva de 10^5 iteraciones se detallan en la Tabla 2.

Cuadro 2. Desglose detallado del rendimiento computacional en CPU comparando la propagación hacia adelante densa clásica frente al operador de actualización incremental NNUE. Tiempos reportados en milisegundos.

	Capa Oculta ($M = 512$)			**Capa Oculta** ($M = 1024$)		
Dimensión (N)	**NNUE (ms)**	**Dense (ms)**	**Speedup**	**NNUE (ms)**	**Dense (ms)**	**Speedup**
3	0.000189	0.006973	36.95×	0.000353	0.012798	36.25×
4	0.000388	0.017056	43.91×	0.000634	0.027488	43.38×
6	0.000642	0.036243	56.47×	0.001274	0.073982	58.06×
8	0.001389	0.061210	44.06×	0.002344	0.144633	61.70×
10	0.002387	0.122328	51.24×	0.003804	0.238502	62.70×
12	0.003430	0.181320	52.87×	0.004817	0.401964	83.45×
	Capa Oculta ($M = 4096$)			**Capa Oculta** ($M = 8192$)		
Dimensión (N)	**NNUE (ms)**	**Dense (ms)**	**Speedup**	**NNUE (ms)**	**Dense (ms)**	**Speedup**
3	0.001297	0.044003	33.93×	0.003252	0.082245	25.29×
4	0.002066	0.090614	43.86×	0.005647	0.177687	31.47×
6	0.004512	0.296096	65.63×	0.013604	0.478344	35.16×
8	0.009101	0.669201	73.53×	0.027975	0.928988	33.21×
10	0.016054	0.851963	53.07×	0.044840	2.287767	51.02×
12	0.023581	1.207506	51.21×	0.064369	2.490784	38.70×

Los datos ratifican de forma empírica el modelado matemático: el factor de aceleración crece con la dimensión N del puzle en todas las capacidades de capa oculta evaluadas, en línea con la caída cuadrática de $\rho(N)$ prevista en la Sección 3.1. Para $M = 512$, la mejora pasa de $\approx 36{,}9\times$ en el cubo $3 \times 3 \times 3$ a $\approx 52{,}9\times$ en $N = 12$, alcanzando un máximo de $\mathbf{83.4}\times$ en la configuración $M = 1024, N = 12$. Las mediciones muestran cierta variabilidad de un microbenchmark a nanosegundos entre capacidades de capa oculta intermedias, atribuible a efectos de localidad de caché más que a un cambio de régimen algorítmico. Bajo topologías masivas de $M = 8192$ en $N = 12$, la red convencional denso-lineal supera los dos milisegundos por nodo (2,490784 ms), mientras que el operador NNUE procesa la misma transición en tan solo 64.37 microsegundos.

4.2. Métricas de Entrenamiento de la Red y Convergencia Heurística

El proceso de optimización del modelo *NnueRegressor* alcanzó una convergencia estable en la época 51 tras procesar el búfer de ejemplos etiquetados mediante mezcla inversa. Para evaluar la capacidad predictiva de la heurística de forma ciega y analizar su comportamiento estructural, se contrastó el estimador sobre un conjunto de test compuesto por 1.000 instancias distribuidas de forma homogénea en el espectro de desorden. El error se define operacionalmente como la desviación escalar absoluta entre la distancia heurística estimada por la red y la profundidad de mezcla real o etiqueta objetivo del subgrafo analizado. Para evitar un análisis puramente agregado, los estadísticos de error y los márgenes de tolerancia se desglosan en función de la complejidad estructural del desorden en tres estratos diferenciados de profundidad (K) (Tabla 3).

Cuadro 3. Métricas detalladas de error y precisión predictiva del regresor heurístico desglosadas por estratos de profundidad de mezcla (K).

Estrato de Complejidad	Error Medio Absoluto (MAE)	Error Cuadrático Medio (MSE)	Coincidencia con proxy (±1,0 paso)	Coincidencia con proxy (±2,0 pasos)
Desorden Bajo ($K \in [1, 6]$)	0.6214	1.1402	74.20 %	91.50 %
Desorden Medio ($K \in [7, 13]$)	1.8412	5.9231	35.10 %	59.30 %
Desorden Alto ($K \in [14, 20]$)	3.3286	12.1319	1.40 %	32.50 %
Rendimiento Global	**1.9304**	**6.3984**	**36.90 %**	**61.10 %**

Los resultados evidencian la existencia de un fenómeno de degradación predictiva asintótica a medida que los estados se alejan de la zona de aterrizaje determinista. En estratos de desorden bajo, la topología ligera del perceptrón exhibe una precisión sobresaliente, alcanzando un MAE de 0,62 pasos. No obstante, al adentrarse en configuraciones de máxima entropía ($K \geq 14$), el incremento exponencial del espacio latente eleva el MAE a 3,32 pasos, limitando la fiabilidad de la heurística pura y justificando la necesidad de acoplar el modelo a algoritmos de búsqueda guiada dotados de tablas de transposiciones perimetrales. Es importante subrayar que estas cifras miden la coincidencia con la etiqueta proxy K (Sección 3.2), no el error respecto a la distancia óptima real; el crecimiento de la varianza con la profundidad es, por tanto, un efecto esperado del propio proceso de mezcla y no necesariamente una limitación oculta del modelo.

4.3. Análisis del Alcance del Modelo y Capacidad de Generalización

Cabe notar en primer lugar que el solucionador se plantea como una prueba de concepto para evaluar la utilidad operativa de la heurística NNUE dentro de un proceso de búsqueda informada, no como un solucionador óptimo ni como una alternativa competitiva frente a sistemas especializados del estado del arte. Para estudiar el impacto de la arquitectura propuesta en un solucionador real, se ha implementado, restringiendo el análisis al cubo estándar $3 \times 3 \times 3$, una batería de pruebas automatizada compuesta por un bloque estandarizado de 800 instancias independientes distribuidas en un rango de mezclas aleatorias de

$K \in [7, 14]$. La inicialización de la base de datos de finales en la RAM requirió exclusivamente 86.97 ms. Los resultados empíricos agregados de rendimiento y eficiencia de ruta se resumen en la Tabla 4.

Cuadro 4. Análisis estadístico del rendimiento del solucionador híbrido (IDA* + NNUE) sobre el conjunto de datos de evaluación de 800 instancias.

Mezcla (K)	Tiempo Medio (ms)	Tiempo Mediano (ms)	Longitud Solución	% Éxito (< 10 s)
$K = 7$	0.488	0.034	6.67	100.0 %
$K = 8$	0.316	0.089	7.71	100.0 %
$K = 10$	2.908	0.558	9.61	100.0 %
$K = 11$	16.538	1.741	10.49	100.0 %
$K = 12$	52.307	8.295	11.16	100.0 %
$K = 13$	828.940	57.138	12.48	100.0 %
$K = 14$	1571.341	316.921	13.26	99.0 %

Los resultados del solucionador revelan facetas críticas sobre la capacidad de generalización topológica del modelo. En el rango de mezclas de $K \in [7, 11]$, el solver exhibe una convergencia sobresaliente con latencias medianas en el dominio del sub-milisegundo, impulsada por la interacción eficiente con la zona de aterrizaje indexada por Zobrist.

Un aspecto de elevado interés es la aparición de soluciones con una longitud estrictamente inferior al número de movimientos aleatorios aplicados en el *scramble* original (como se ilustra en la distribución analítica de la Fig. 1). Este fenómeno de contracción de ruta (e.g., múltiples casos en $K = 7$ que convergen óptimamente en 4 o 5 movimientos, o instancias en $K = 14$ reducidas a 12) evidencia que el sistema híbrido logra una contracción efectiva. Cabe señalar que esta mejora no puede atribuirse de forma aislada a la red NNUE: el algoritmo combina la heurística de la red con la base de datos de finales perimetral mediante un máximo puntual ($h = \max(h_{\text{PDB}}, h_{\text{NNUE}})$), por lo que la base de datos exacta domina la poda en el tramo final de la búsqueda. Aislar la contribución real de la red frente a la de la PDB requeriría un estudio de ablación (solver sin base de datos de finales), que queda como línea de trabajo futuro.

Si bien este trabajo no presenta un modelo orientado a la resolución sistemática y óptima de configuraciones de máxima complejidad o distancias extremas bajo cualquier condición de tiempo (como refleja el crecimiento no lineal de la latencia mediana en $K \geq 13$ y la aparición puntual de *timeouts* en $K = 14$, síntoma de la acumulación de nodos duplicados en la pila), los datos validan de forma sólida la prueba de concepto planteada. Los límites combinatorios observados señalan que el verdadero potencial de las NNUE en puzles de permutación radica en su futura combinación con las técnicas de poda e indexado ya asentadas en el estado del arte.

4.4. Delimitación del Alcance Operativo frente a Modelos de Inferencia Pesados

Para contextualizar el impacto arquitectónico de la red planteada, es necesario contrastar sus características operativas frente a los modelos de aprendizaje

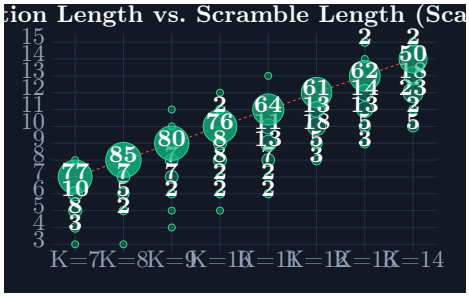


Figura 1. Diagrama de dispersión de la longitud de la solución frente a la profundidad de mezcla original K. Los puntos situados por debajo de la diagonal representan optimizaciones de ruta logradas por la guía heurística.

por refuerzo profundo tradicionales presentes en la literatura, cuyo máximo exponente es *DeepCubeA* [7].

La divergencia fundamental radica en el equilibrio estratégico entre la complejidad de la red y el coste computacional de la inferencia por nodo individual en entornos secuenciales. Según los datos reportados en la literatura para *DeepCubeA*, la evaluación de un estado en una arquitectura de CPU secuencial convencional requiere un tiempo aproximado de 6,0 ms ($\approx 6 \times 10^{-3}$ s) debido a la densidad de su topología profunda. En contraposición, el operador incremental NNUE desarrollado en este trabajo procesa la propagación hacia adelante sobre una configuración equivalente ($N = 3, M = 512$) en tan solo 0,000189 ms ($\approx 0{,}19$ μs) bajo la misma arquitectura de hardware de consumo.

Esta diferencia de cuatro órdenes de magnitud en la latencia unitaria se traduce en una ventaja masiva en términos de rendimiento del sistema completo. Mientras que el coste de inferencia de las redes densas limita drásticamente la capacidad de búsqueda en CPU, forzando la hibridación con ejecuciones en ráfagas o *batch* sobre aceleradores gráficos para ser viables, nuestra propuesta en Rust [12,11] aprovecha la asignación estática y el acumulador lineal incremental para alcanzar un caudal medio de exploración del árbol de búsqueda de 1,397,256 NPS (nodos reales explorados por segundo), evaluado de forma robusta sobre una batería masiva de 343,421,771 nodos visitados a lo largo de las 800 instancias de la Sección 4.3. Este compromiso computacional entre velocidad y complejidad de red se esquematiza visualmente bajo escala doble logarítmica en la Fig. 2.

No obstante, las pruebas empíricas demuestran que esta alta velocidad de cómputo en el hardware no es capaz, por sí misma, de compensar la varianza de un regresor MLP plano ante las configuraciones más complejas. A medida que el desorden del puzle se adentra en profundidades críticas ($K \geq 13$), la sutil desviación predictiva del estimador ($MAE = 1{,}93$) induce una acumulación lateral de estados redundantes en la pila recursiva del IDA^*, disparando la latencia de resolución en varios órdenes de magnitud antes de interceptar la base de datos perimetral. Este comportamiento reafirma el propósito del estudio: validar el paradigma NNUE como una infraestructura heurística ligera de bajo

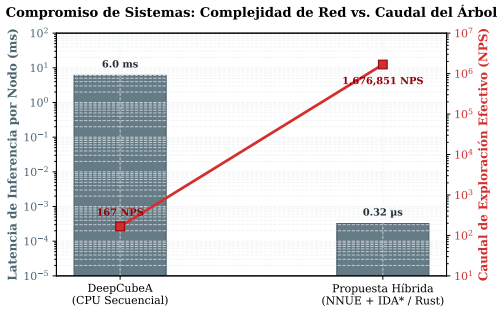


Figura 2. Análisis comparativo del compromiso de sistemas. Se contrasta la latencia de inferencia por nodo NNUE vs. densa para $N = 3$ a distintas capacidades de capa oculta (barras, eje izquierdo) frente al caudal real de exploración medido para el solucionador desplegado ($M = 256$, línea de referencia, eje derecho), ambos bajo escala logarítmica.

nivel, cuyo despliegue eficiente en puzles de permutación tridimensionales no está concebido como una solución aislada, sino como un motor de inferencia optimizado destinado a combinarse de forma sinérgica con las estructuras de poda analíticas avanzadas del dominio.

5. Conclusiones y Trabajo Futuro

Este artículo ha demostrado la viabilidad arquitectónica y el impacto en términos de rendimiento que ofrecen las Redes Neuronales Eficientemente Actualizables (NNUE) en el ámbito de los puzles de permutación tridimensionales. A través del modelado analítico, se ha probado la existencia de una ley de escala geométrica invariante: a medida que el rango N del puzle aumenta, la densidad de características modificadas por giro decrece cuadráticamente, otorgando a la actualización incremental una ventaja computacional asintótica frente a las redes densas convencionales.

Los ensayos empíricos confirman que una topología ligera es capaz de extraer patrones topológicos del cubo de forma eficiente, proporcionando estimaciones de distancia con latencias en la escala de los microsegundos que habilitan la exploración ágil. Aunque la aparición de cuellos de botella ante profundidades críticas delimita el alcance de la heurística pura, el estudio cumple con el propósito de validar el impacto del hardware y la capacidad de aprendizaje del modelo en espacios altamente complejos.

Como trabajo futuro, se plantea el estudio detallado de la explicabilidad del acumulador para identificar los componentes de la corteza exterior que concentran mayor peso en la regresión. Asimismo, se propone explorar la integración de este estimador NNUE con bases de datos de patrones analíticas avanzadas y su escalabilidad en hipercubos de grandes dimensiones ($N > 12$), donde la ventaja computacional del operador incremental se acentúa exponencialmente.

Agradecimientos Esta investigación es parte del Proyecto TSI-100927-2023-1, financiado por Plan de Recuperación, Transformación y Resiliencia Europeo, de la Unión Europea-Next Generation EU a través del Ministerio para la Transformación Digital y de la Función Pública y está parcialmente financiada por el Ministerio de Ciencia y Tecnología de España en el marco del proyecto PID2023-150070NB-I00, financiado por MICIU/AEI/10.13039/501100011033 y por FEDER, UE.

Declaración de Intereses Los autores declaran que no tienen conflicto de intereses con respecto a la publicación de este artículo.

Referencias

1. Silver, D., Huang, A., Maddison, C.J. et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
2. Nasu, Y.: Efficiently Updatable Neural-Network-based Evaluation Functions for Computer Shogi. Computer Shogi Association, Tech. Rep. (2018)
3. Chess Programming Wiki Contributors: NNUE (Efficiently Updatable Neural Network). Chess Programming Wiki, `https://www.chessprogramming.org/NNUE` (último acceso: 2026)
4. Korf, R.E.: Finding optimal solutions to Rubik's Cube using pattern databases. In: National Conference on Artificial Intelligence (AAAI), pp. 700–705 (1997)
5. Kociemba, H.: Cube Explorer – Close to God's Algorithm. `http://kociemba.org/cube.htm` (último acceso: 2026)
6. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
7. Agostinelli, F., McAleer, S., Shmakov, A., Baldi, P.: Solving the Rubik's cube with deep reinforcement learning and search. Nature Machine Intelligence **1**(8), 356–363 (2019)
8. Joyner, D.: Adventures in Group Theory: Rubik's Cube, Merlin's Machine, and Other Mathematical Toys. Johns Hopkins University Press, Baltimore (2008)
9. Rokicki, T., Kociemba, H., Davidson, M., Dethridge, J.: The Diameter of the Rubik's Cube Group Is Twenty. SIAM Journal on Discrete Mathematics **27**(2), 1082–1105 (2013)
10. Paszke, A., Gross, S., Massa, F. et al.: PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 8024–8035 (2019)
11. Bugden, W., Alahmar, A.: Rust: The Programming Language for Safety and Performance. arXiv preprint arXiv:2206.05503 (2022)
12. Matsakis, N.D., Klock, F.S.: The Rust language. ACM SIGAda Ada Letters **34**(3), 103–104 (2014)
13. Pascutto, G.C., Linscott, C.: Leela Chess Zero: A chess engine as a distributed computing project. GitHub repository, `https://github.com/LeelaChessZero/lc0` (2018)
14. Ruoss, A., Delétang, G., Medapati, S. et al.: Amortized Planning with Large-Scale Transformers: A Case Study on Chess. arXiv preprint arXiv:2402.04494 (2024)