# CÓMO MEJORAR LA OPTIMIZACIÓN DE ESQUEMAS DE RECARGA CON NEUTRONET: EXPERIENCIA Y FUNCIÓN MULTIOBJETIVO

JAVIER ORIVE SOTO
Ingeniero de diseño nuclear PWR
ENUSA INDUSTRIAS AVANZADAS, SA, SME.

JUAN ANDRÉS LOZANO MONTERO
Ingeniero de diseño nuclear PWR
ENUSA INDUSTRIAS AVANZADAS, SA, SME.

## INTRODUCCIÓN

### Contexto

La gestión del combustible nuclear *in-core* se refiere a las técnicas y estrategias utilizadas para optimizar el uso del combustible dentro del núcleo de un reactor nuclear durante su operación. Dentro de esta gestión, la búsqueda de Esquema de Recarga (ER) es un proceso fundamental, el cual tiene por objetivo optimizar la elección del combustible fresco y la distribución de todos los elementos combustibles dentro del núcleo con el fin de cumplir los requisitos energéticos y operacionales de la planta, manteniendo todas las garantías de seguridad. Tradicionalmente, este ha sido un proceso intuitivo e iterativo, basado principalmente en la experiencia, en la que manualmente se evalúan distintos ER haciendo uso de un código neutrónico hasta encontrar el núcleo que se ajuste mejor a los requisitos energéticos y operacionales de la planta.

La principal dificultad radica en el enorme número de combinaciones posibles en núcleos de reactores comerciales, como es el caso en los PWR de 3 lazos con 157 elementos combustibles dentro de su núcleo. En este tipo de núcleos, el número de combinaciones tiene del orden de entre $10^{20}$ y $10^{100}$ dependiendo de la simetría y restricciones utilizadas en la búsqueda. El hecho de que el proceso de búsqueda se defina como un problema de optimización discreta (las posiciones de los elementos combustibles son variables de entrada discretas a la función de optimización) y que la respuesta neutrónica sea altamente no-lineal y dependiente de fenómenos locales, hace que no exista función analítica a optimizar, lo cual hace que sea altamente complejo hallar óptimos globales.

En los últimos años ENUSA ha estado desarrollando la herramienta Neutronet (véase [1] y [2]), capaz de calcular soluciones al problema automáticamente y con un criterio de búsqueda inteligente. Esto permite reducir significativamente el esfuerzo requerido, en términos de horas-hombre, y acortar los plazos necesarios en esta fase del diseño. Además, ofrece la posibilidad de encontrar soluciones que se alejen de la experiencia previa del diseñador nuclear, ampliando así el espacio de alternativas consideradas.

En consecuencia, esta herramienta resulta especialmente útil para rediseños de emergencia, ciclos con cambios significativos respecto a la experiencia anterior y para estudios multiciclos que requieren la evaluación de distintas alternativas en poco tiempo.

### Estado de la técnica

Existe un gran interés en el desarrollo de soluciones que hacen uso de técnicas de optimización para resolver el problema, sobre todo en el campo de la investigación. Dentro del campo de la optimización existe una gran variedad de algoritmos que pueden ser utilizados para una multitud de problemas dependiendo de su tipo y complejidad. Entre las técnicas que han demostrado utilidad en la optimización de esquemas de recarga (ER) se encuentran el *Simulated Annealing* (SA), los Algoritmos Genéticos (GA) y el *Tabu Search* (TS), tal como se recoge en las referencias [3], [4] y [5] respectivamente. Todos estos algoritmos tienen en común que son del tipo heurístico, lo cual significa que son capaces de buscar soluciones aproximadas mediante estrategias de búsqueda inteligentes, sin necesidad de explorar exhaustivamente todo el espacio de combinaciones. Como consecuencia de no explorar todo el espacio de combinaciones, no se asegura la obtención del óptimo global.

En el desarrollo de Neutronet se ha optado por la implementación de un algoritmo genético, dado su buen desempeño en la exploración de soluciones, permitiendo encontrar configuraciones óptimas de recarga en reactores PWR de forma automatizada y con menor esfuerzo computacional.

## OPTIMIZACIÓN DE LA BÚSQUEDA DE ESQUEMA: NEUTRONET

Como se ha explicado anteriormente, Neutronet, programado en Python, implementa un optimizador heurístico basado en un algoritmo genético.

La arquitectura del programa se compone de los siguientes elementos:

- **Utilidades y funciones auxiliares.** Son funciones que permiten la lectura, volcado y transformación a Python de los datos de partida contenidos en los archivos del código nodal de un núcleo de referencia, para su posterior uso en el programa. También implementan una transformación reversa que permite volcar los datos de Python para la generación de los archivos *input* del código neutrónico.
- **Optimizador.** Se basa en un algoritmo genético y es el núcleo del programa. Este tipo de programa iterativo genera nuevos ER partiendo de mezclar los mejores ER del paso de iteración previo. Esto permite ir mejorando paulatinamente los esquemas candidatos.
- **Predictor/Simulador.** Es la parte del optimizador que se encarga de simular o predecir distintas variables de salida partiendo de los ER propuestos. Para ello, o bien puede utilizarse el código neutrónico (ANC8) para la simulación de la operación del ciclo del ER en estudio o un predictor basado en una red neuronal entrenada para este cometido. Esta es la etapa que conlleva un mayor tiempo y esfuerzo computacional, en especial, si se usa la opción del código neutrónico.
- **Evaluador.** Es la parte del código que evalúa asignando una puntuación a los distintos candidatos para su comparación. Para ello, implementa una función objetivo o multiobjetivo que toma distintas variables de los resultados del predictor/simulador y le asigna un valor numérico.

### Optimizador

El optimizador es el núcleo del programa, ya que, partiendo de un ER de referencia, se encarga de buscar ER que cumplan con los criterios técnicos establecidos, maximizando el rendimiento del núcleo y respetando las restricciones impuestas.

En Neutronet el optimizador implementa una variante de un algoritmo genético. Este tipo de algoritmo se clasifica dentro de la categoría de los algoritmos evolutivos poblacionales, lo que significa que parte de una diversidad de soluciones iniciales y, mediante iteraciones en las que se modifica la población, mejora progresivamente las soluciones obtenidas. Dado el gran número de posibles configuraciones, el proceso es estocástico, lo que implica que los resultados pueden variar entre ejecuciones.

A continuación, se enumeran los pasos que conforman el optimizador:

1. **Inicialización:** Este proceso es ejecutado únicamente en el primer paso de la iteración y crea una población inicial de posibles ER. Este proceso puede ser ajustado para que los ER sean totalmente aleatorios (generando combinaciones aleatorias de

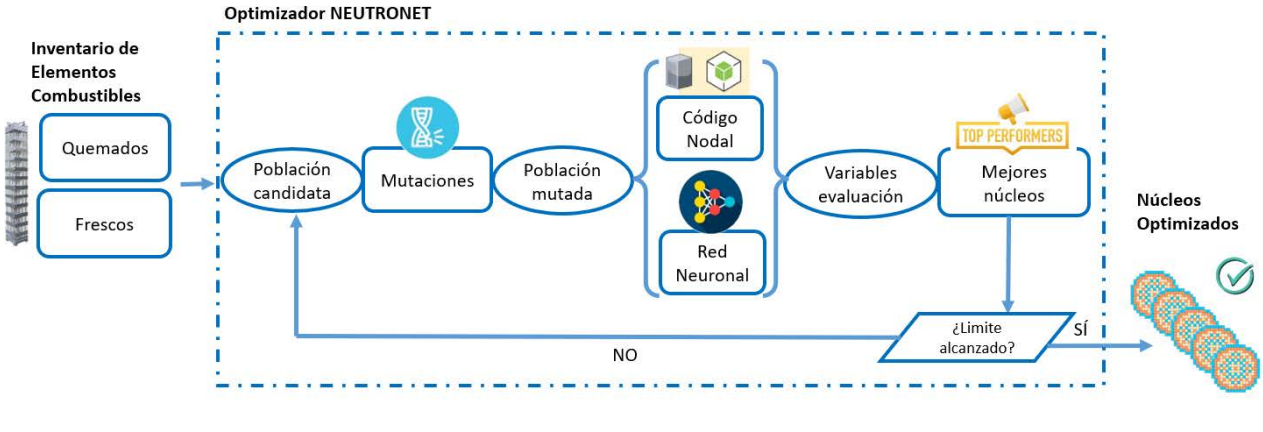


**Figura 1.** Esquema de la arquitectura de Neutronet.

los elementos combustibles del núcleo de partida) o controlado (realizando pequeñas variaciones al ER de partida)

2. **Evaluación:** Se obtienen las distintas variables características del ciclo para cada ER, como por ejemplo la duración del ciclo, la curva de boro y los factores de pico de potencia de los elementos combustibles. Estos valores pueden obtenerse tanto mediante simulación con el código nodal como mediante inferencia realizada por la red neuronal entrenada ad hoc para esta aplicación. En el apartado 2.2 se explican con mayor detalle ambos métodos. Tras obtener los valores de las variables características del ciclo, los ER son evaluados según una función objetivo *(fitness)*, que mide la bondad de la solución. En el apartado 2.3. se detallan las funciones objetivo disponibles para el proceso de evaluación.
3. **Selección:** Tras evaluar los distintos ER propuestos, se ordenan en base a la puntuación otorgada por la función objetivo y se seleccionan los mejores individuos para reproducirse.
4. **Cruce** *(crossover)*: Se combinan partes de los ER seleccionados para crear nuevos ER. Este proceso se asemeja a la reproducción biológica y es el motivo por el que se conoce al algoritmo como genético. Durante el proceso dos ER son combinados, considerando la restricción de que los elementos combustibles no pueden repetirse, generando así un nuevo ER.
5. **Mutación:** Se introducen pequeñas modificaciones aleatorias para mantener la diversidad genética. Para ello, se realizan pequeñas traslaciones aleatorias en las posiciones de varios elementos combustibles. Esto retrasa ligeramente el proceso de convergencia, pero incrementa el área de búsqueda, y aumentando la probabilidad de encontrar soluciones diferentes.
6. **Reemplazo:** Se conforma una nueva generación de ER, y el proceso se repite desde el punto 2 hasta que se complete el número de iteraciones o se alcance un criterio de búsqueda previamente establecido.

Tras finalizar el proceso optimización, y como característica de los algoritmos poblacionales, se obtiene múltiples ER candidatos, clasificados por puntuación. El número de candidatos depende del número de candidatos supervivientes establecido para cada iteración. Por tanto, como ventaja de este tipo de algoritmo, se pueden obtener una gran cantidad de posibles candidatos para ser analizados en mayor detalle, lo cual mejora la probabilidad de encontrar soluciones innovadoras.

Además, durante el desarrollo de Neutronet, con el fin de facilitar el proceso de generación de nuevos ER durante el cruce y mutación, se ha implementado la posibilidad de utilizar un mapa con restricciones de posición de núcleo. Este mapa permite definir, para cada posición del núcleo, el tipo de elemento combustible según su número de ciclos: fresco, de un ciclo o de dos ciclos. En consecuencia, se reduce el espacio de búsqueda, se mejora la convergencia del optimizador y se garantiza que los esquemas propuestos cumplan con otros criterios del esquema.

**Predictor/Simulador**

Una vez se dispone de los ER propuestos por el optimizador, es necesario calcular los parámetros del ciclo que se usan habitualmente para la selección de ER, tales como la curva de boro, los factores de pico de potencia, el quemado máximo, entre otras. Para ello, se requiere simular el comportamiento del núcleo bajo cada configuración de ER.

El método principal utilizado para esta simulación es el código nodal ANC, que es el código licenciado en ENUSA para los estudios de diseño de recargas PWR. ANC permite obtener resultados precisos y validados, pero su uso implica un mayor coste computacional. Cada simulación de un ER requiere del orden de un minuto, lo que se multiplica significativamente al evaluar miles de esquemas en un proceso de optimización.

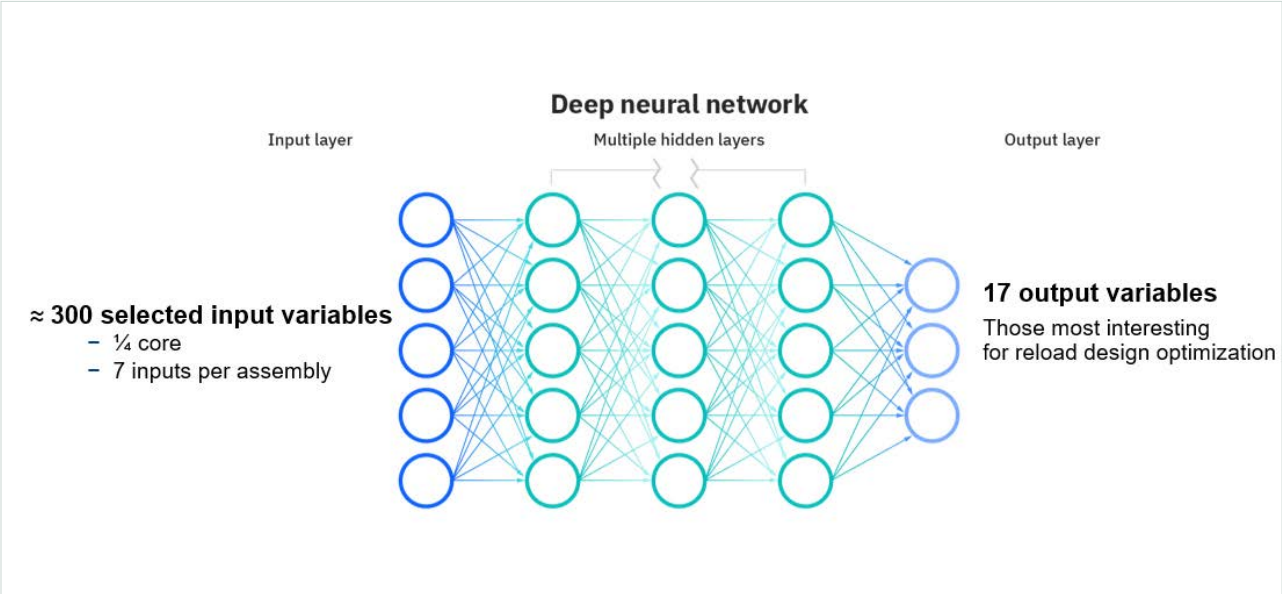


**Figura 2.** Estructura simplificada de la red neuronal MLP.

Con el objetivo de reducir este tiempo de cálculo, en la primera fase del proyecto se desarrolló una red neuronal artificial entrenada específicamente para predecir 17 variables relevantes del núcleo, partiendo únicamente de los datos de entrada del combustible y la configuración del ER (véase **Figura 2**). Esta red fue presentada en trabajos anteriores (véase [1] y [2]) y mostró un rendimiento prometedor en fases iniciales del proyecto. Sin embargo, debido a cambios recientes en las estrategias de carga utilizadas en las últimas recargas, el rendimiento de la red neuronal se vio afectado, disminuyendo su capacidad predictiva. Por este motivo, en la versión actual de Neutronet se ha optado por utilizar exclusivamente la opción de Neutronet con el código neutrónico ANC para la evaluación de los ER, garantizando así la fiabilidad de los resultados obtenidos.

**Evaluador: funciones objetivo y multiobjetivo**

El evaluador es el componente encargado de valorar la calidad de cada solución propuesta por el optimizador, según criterios definidos por funciones objetivo. Su función principal es asignar una puntuación que permita comparar distintas configuraciones del núcleo y guiar la evolución hacia soluciones más adecuadas.

Para ello, se han implementado tanto funciones objetivo-simples, que clasifican las soluciones en función de un único parámetro, como funciones multiobjetivo, que permiten evaluar simultáneamente varias variables relevantes. A continuación, se detallan las distintas funciones simples implementadas y que pueden ser seleccionadas por el usuario de Neutronet:

- **Pico de potencia.** Esta función toma el valor del pico de potencia máximo y tiene por objetivo su minimización.
- **Quemado/Longitud de ciclo.** Esta función tiene por objetivo maximizar la duración del ciclo.
- **Concentración de boro inicial.** Esta función esta implementada para la reducción de la concentración de boro máxima al inicio del ciclo ya que es crítica en algunos accidentes.

Asimismo, también se ha probado la implementación de funciones multiobjetivo. Esta última variante es más compleja, ya que cada parámetro puede tener rangos de magnitud muy distintos, lo que dificulta asignar pesos adecuados en una única función de evaluación. En este caso, la función implementada busca minimizar los picos de potencia del núcleo y maximizar el quemado de ciclo simultáneamente. La función utilizada se muestra a continuación:

$$Score_{total} = f_1(\text{PPF}) + w \cdot f_2(\text{BU})$$

Dónde el primer término del score representa una penalización o bonificación discreta basada en el valor máximo de factor de pico calculado, y el segundo término está inversamente relacionado con la longitud máxima del ciclo, incentivando soluciones con mayor quemado. Dentro del segundo término los valores numéricos impuestos buscan asemejar la magnitud del aporte del quemado del ciclo a la escala de los parámetros definidos para el factor de pico, permitiendo que ambos factores (quemado y factores de pico máximos) tengan un peso comparable en la evaluación total.

## RESULTADOS Y EXPERIENCIA DE USO

Para comprobar el funcionamiento de Neutronet, se han realizado múltiples ejecuciones con distintas configuraciones de hiperparámetros. Se han realizado variaciones tanto al número de generaciones y tamaño de la población, con el fin de ver el efecto del número de ER simulados, como al tipo de función objetivo. Estas ejecuciones permiten evaluar la robustez del algoritmo y su capacidad de adaptación a diferentes criterios de optimización.

Además, en paralelo a las ejecuciones automáticas, se han llevado a cabo búsquedas manuales de esquemas de recarga, siguiendo el procedimiento tradicional utilizado en ENUSA. Este enfoque comparativo ha sido clave para validar los resultados obtenidos por Neutronet, contrastando su rendimiento frente a la experiencia acumulada en el diseño de esquemas de recarga.

En la **Tabla I** se presenta una muestra de los resultados obtenidos para seis ejecuciones de un mismo núcleo en Neutronet, variando exclusivamente los hiperparámetros del programa. Se pueden apreciar los siguientes puntos:

- Al establecer el pico de potencia máximo como parámetro a optimizar, se consiguió alcanzar valores de 1,469, significativamente por debajo del límite de 1,5 y de los valores usuales obtenidos en diseño (los cuales rondan 1,48), lo que indica una buena estabilidad del algoritmo.
- En el caso en el que el quemado máximo se estableció como objetivo, se consiguió alargar el ciclo en unos

| Hiperparámetros de Neutronet | | | Resultados | | | |
|---|---|---|---|---|---|---|
| Nº de generaciones | Población | Función objetivo | Pico de potencia [-] | Boro inicial [ppm] | Quemado de ciclo [MWd/tU] | Tiempo [h:m:s] |
| 50 | 120 | P. potencia | 1,469 | 1522 | 19918 | 6:36:32 |
| 50 | 50 | P. potencia | 1,469 | 1523 | 19928 | 4:35:23 |
| 30 | 120 | P. potencia | 1,469 | 1523 | 19907 | 4:13:58 |
| 50 | 120 | Quemado | 1,494 | 1524 | 19953 | 8:39:41 |
| 30 | 50 | P. potencia-Quemado | 1,473 | 1524 | 19942 | 4:26:36 |
| 30 | 120 | Boro inicial | 1,473 | 1515 | 19819 | 3:56:52 |

**Tabla 1.** Resultados ejecución de Neutronet.

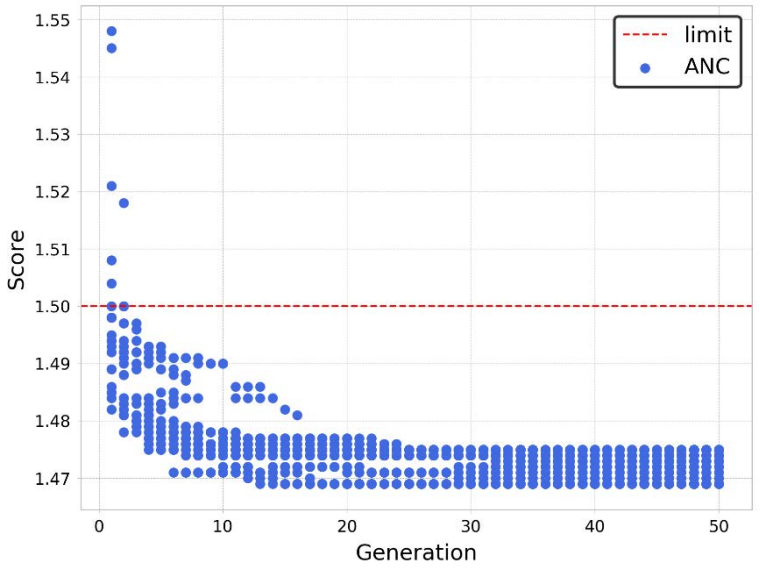


**Figura 3.** Ejemplo del rendimiento de una ejecución de optimización del pico máximo de potencia.

20-30 MWd/tU, aunque los factores de pico se vieron empeorados.

- La concentración de boro inicial de la mayoría de los casos se mantuvo en torno a los 1523 ppm. En cambio, para el caso en el que se estableció este parámetro como objetivo, se pudo reducir su valor hasta as 1515 ppm, a expensas de empeorar ligeramente el quemado máximo de ciclo.
- En la ejecución del caso con la función multiobjetivo pico de potencia-quemado, se obtuvo un buen compromiso en el proceso de optimizar ambos parámetros.
- El tiempo de ejecución depende mayoritariamente del número de ER simulados, es decir, del número de generaciones por el número de ER que conforma la población. Este tiempo osciló entre las 4h y 8h 30min, lo cual ronda los 40 segundos por ER simulado, tomando que se pueden llevar a cabo 10 ejecuciones del código neutrónico en paralelo.
- Debido a que el proceso de convergencia de los algoritmos genéticos es bastante rápido, un número de 30 generaciones es suficiente para alcanzar una buena solución. Por lo que el aumento del número de generaciones a valores de 50 no es necesario, ya que aumenta significativamente el tiempo necesario para terminar la ejecución.

La **Figura 3** muestra el rendimiento típico de una ejecución de Neutronet. Este tipo de comportamiento es característico de los algoritmos genéticos. Al inicio del proceso, se parte de una población con ER de baja calidad, pero con una gran diversidad. Esta diversidad inicial permite una mejora sustancial en las primeras generaciones, ya que el algoritmo explora activamente el espacio de soluciones. Conforme se avanza en las iteraciones (a partir de la generación 20) se alcanzan configuraciones cercanas a un mínimo local y el algoritmo tiende a estancarse. Esto se debe a que la diversidad de los candidatos disminuye progresivamente, lo que limita la capacidad de encontrar soluciones significativamente mejores. Este comportamiento es deseable en la fase final del proceso, ya que indica que el algoritmo ha identificado una región del espacio de soluciones con alto rendimiento. Sin embargo, también pone de manifiesto la importancia de mantener mecanismos de mutación y diversidad para evitar una convergencia prematura.

## CONCLUSIONES

En este trabajo se expone el desarrollo de Neutronet, una herramienta capaz de encontrar esquemas de recarga que cumplan mejor los criterios de búsqueda mediante la aplicación de técnicas de inteligencia artificial, concretamente algoritmos genéticos acoplados a redes neuronales. Esto permite automatizar el proceso de búsqueda de esquemas de recarga, reduciendo significativamente el esfuerzo manual y ampliando el espacio de soluciones exploradas.

Una de las principales ventajas de Neutronet es la posibilidad de seleccionar distintas funciones objetivo, adaptando el criterio de evaluación a las necesidades específicas del estudio. Las funciones simples han demostrado un rendimiento muy robusto, proporcionando resultados consistentes y de alta calidad. La función multiobjetivo, aunque prometedora, requiere un ajuste más fino de los pesos asignados a cada parámetro. Para ello, se pretende utilizar la experiencia acumulada en su uso como base para una futura calibración más precisa. En general, Neutronet ha supuesto una gran ayuda en el proceso de búsqueda de ER y está siendo utilizado regularmente en los últimos diseños.

De cara al futuro, se plantea retomar la línea de desarrollo de la red neuronal, entrenándolo con un conjunto de datos más amplio y con valores más adaptados a las estrategias actuales de carga. Esta mejora permitiría

acelerar significativamente el proceso de evaluación de esquemas, reduciendo la dependencia del código nodal y facilitando la exploración de candidatos innovadores con mayor rapidez.

## AGRADECIMIENTOS

Deseamos expresar nuestro más sincero agradecimiento a Carles Mesado, Alejandro Carrasco y Javier Riverola, que fueron los primeros en impulsar el proyecto Neutronet y tuvieron un rol esencial en su progreso. Su esfuerzo, dedicación y visión han sido esenciales para transformar esta herramienta en algo efectivo y útil. También queremos dar las gracias al equipo de diseño nuclear por su voluntad de ayudar en la validación de la herramienta y por sus valiosos consejos y comentarios.

## REFERENCIAS

[1] A. Carrasco, C. Mesado, «Neutronet: Optimización de la Búsqueda de Esquema de Recarga mediante Inteligencia Artificial», *47ª Reunión Anual de la Sociedad Nuclear Española*, Cartagena, 2022.

[2] A. Carrasco, C. Mesado, «Neutronet: Loading Pattern Optimization using Machine Learning techniques», *Proceedings of the European Nuclear Young Generation Forum ENYGF23, Krakow, 2023*.

[3] J. G. Stevens, K. S. Smith, K. R. Rempe & T. J. Downar, «Optimization of Pressurized Water Reactor Shuffling by Simulated Annealing with Heuristics», *Nuclear Science and Engineering*, 121:1, 67-88, 1995, DOI: 10.13182/NSE121-67.

[4] P.W. Poon, G.T. Parks, «Application of genetic algorithms to in-core nuclear fuel management optimization » *Mathematical methods and supercomputing in nuclear applications Proceedings* Vol 1, (p. 878). Germany, 1993.

[5] C. Lin, J. Yang, K. Lin & Z. Wang, «Pressurized Water Reactor Loading Pattern Design Using the Simple Tabu Search», *Nuclear Science and Engineering*, 129:1, 61-71, 1998, DOI: 10.13182/NSE98-A1963. ■