Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN CIENCIA E ENXEÑARÍA DE DATOS

Optimización del problema de *scheduling* en sistemas de comunicación Cell Free Massive MIMO mediante Deep Contextual Bandit

Estudante: Marta Benavente Vilas

Dirección: Óscar Fresnedo Arias

Dariel Pereira Ruisánchez

A Coruña, junio de 2025.

A mi familia

Agradecimientos

En primer lugar, quiero agradecérselo a mis tutores, Óscar y Dariel, por su orientación y ayuda constante a lo largo de todo el proyecto. Ha sido un proceso en el que he crecido tanto a nivel académico como personal, en gran parte ha sido gracias a ellos.

A mi familia, mi ejemplo y mi inspiración diaria para seguir creciendo. Gracias por ser la base y los pilares que me han reforzado en cada paso y por confiar en mí incluso más de lo que yo misma lo hacía. Sin ellos, esto no habría sido posible.

Finalmente, a mis amigos del grado, gracias por estar siempre ahí, por alegrarme los días y por hacer que estos años hayan pasado volando.

Resumen

En el mundo actual, donde la conectividad es fundamental, maximizar el rendimiento de las redes inalámbricas es clave para garantizar eficiencia y calidad del servicio. Sin embargo, los sistemas actuales basados en celdas presentan limitaciones, ya que la señal disminuye significativamente a medida que los usuarios se alejan de la antena principal, especialmente en la periferia de las celdas. Frente a este desafío, surge el enfoque *Massive Cell-Free*, que elimina la dependencia de celdas al distribuir puntos de acceso de manera uniforme en el escenario. Este proyecto aborda el problema de la asignación de recursos (*scheduling*) en este tipo de redes mediante el uso de modelos de aprendizaje automático, optimizando la asignación de puntos de acceso a usuarios para garantizar una conexión de calidad independientemente de su ubicación espacial. A partir de escenarios simulados, se obtuvieron variables clave como la ganancia y la interferencia de los usuarios respecto a cada punto de acceso, las cuales se utilizaron para entrenar el modelo principal: una red basada en *Deep Contextual Bandit* (DBC) con una función de recompensa basada en la calidad de la señal recibida. Durante el desarrollo surgieron otras aproximaciones complementarias, como modelos basados en pérdida y enfoques de *clustering*, que fueron evaluadas frente al modelo de DCB y una asignación aleatoria como línea base. Los resultados muestran que los modelos basados en pérdida y el de *Deep Contextual Bandit* proporcionan una asignación más eficiente y dinámica, optimizando la calidad de la conexión y superando las limitaciones de los métodos tradicionales. Esta solución representa un avance prometedor para redes inalámbricas avanzadas como Beyond 5G (B5G) y 6G, contribuyendo a una conectividad más robusta y flexible.

Abstract

In today's world, where connectivity is essential, maximizing the performance of wireless networks is key to ensuring efficiency and service quality. However, current cell-based systems face significant limitations, as signal strength decreases considerably as users move away from the main antenna, particularly at the cell edges. To address this challenge, the Massive Cell-Free approach eliminates cell dependency by distributing access points uniformly across the scenario. This project tackles the resource allocation problem (scheduling) for this type of networks through machine learning models, optimizing the assignment of access points to users to ensure high-quality connections regardless of spatial location. From simulated scenarios, key variables such as user gain and interference with respect to each access point were extracted and used to train the main model: a Deep Contextual Bandit network with a

reward function based on the total received signal quality. Throughout the development process, complementary approaches arised, including loss-based models and clustering methods, which were evaluated against the Deep Contextual Bandint model and a baseline random assignment approach. Results show that the loss-based and Deep Contextual Bandit models provide a more efficient and dynamic allocation, optimizing connection quality and overcoming the limitations of traditional methods. This solution represents a promising advance for next-generation wireless networks such as Beyond 5G (B5G) and 6G, contributing to more robust and flexible connectivity.

Palabras clave:

- Cell-Free Massive MIMO
- Asignación de Recursos (*Scheduling*)
- Deep Contextual Banidts
- Modelos Basados en Pérdida
- Clustering Difuso
- Asignación de Puntos de Acceso
- Aprendizaje Máquina
- Redes Inalámbricas

Keywords:

- Cell-Free Massive MIMO
- Resource Allocation (*Scheduling*)
- Deep Contextual Bandits
- Loss-Based Models
- Fuzzy Clustering
- Access Point Assignment
- Machine Learning
- Wireless Networks

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la era de la información, las redes de comunicación se han asentado como los cimientos del ecosistema digital, en el cual estar conectado es un requisito esencial tanto en el ámbito social como en el profesional. Sin embargo, la expansión de dispositivos conectados y el crecimiento exponencial del tráfico de datos, impulsados por la digitalización y la interconectividad global, han generado una demanda creciente de conexiones de mayor calidad. Aplicaciones de alta exigencia, como la realidad aumentada o la telemedicina, requieren infraestructuras de comunicación más robustas y eficientes para garantizar un rendimiento óptimo. Este desfase entre la demanda de los UEs y la capacidad de las redes actuales ha impulsado la búsqueda de soluciones innovadoras que permitan mejorar la eficiencia y calidad del servicio. Todos estos factores han motivado que las futuras generaciones de datos móviles (Beyond 5G y 6G) demanden una gran variedad de requisitos en términos de calidad de servicio muy complejos y exigentes, entre los que destacan un soporte masivo de UEs conectados de forma simultánea, altas velocidades de transmisión, latencias muy bajas, calidad de conexión homogénea independientemente de la localización, consumo de energía razonable, etc.

Los sistemas de comunicación basados en *Cell-Free Massive Multiple-Input Multiple-Output* (MIMO) surgen como una alternativa prometedora en este contexto, rompiendo con la arquitectura tradicional de los sistemas basada en celdas. Esta aproximación consiste fundamentalmente en distribuir una gran cantidad de elementos de conexión mucho más sencillos a lo largo del área de cobertura, los cuales cooperan entre sí para dar servicio a los UEs. Al descentralizar la infraestructura, se evita la degradación de la señal en la periferia de las celdas, lo que a su vez se traduce en una mejora de la cobertura, una mayor eficiencia espectral, una calidad de servicio más homogénea y una mayor flexibilidad para adaptarse a entornos de comunicación dinámicos debido a la movilidad de los UEs. Este proyecto se centra en optimizar estos sistemas abordando el problema del *scheduling*[1] a través de diversos modelos de

[1] Se refiere a la asignación eficiente de recursos de red a múltiples UEs, optimizando el rendimiento global

aprendizaje automático, entre ellos Deep Contextual Bandits (DCB) y enfoques de aprendizaje basados en pérdida, dos estrategias prometedoras para llevar a cabo una asignación eficiente de los recursos. Esta iniciativa permite un reparto más inteligente y adaptable de los recursos de la red, constituyendo un avance para las redes inalámbricas avanzadas.

1.1 Objetivos

El principal objetivo de este proyecto es el desarrollo de un modelo de DCB que permita la asignación eficiente de punto de accesos (APs) a dispositivo usuarios (UEs)[2] usando como recompensa el *sum-rate*[3] de estos últimos. Esto resulta fundamental para satisfacer de forma eficiente los requisitos tan exigentes demandados por las nuevas generaciones de datos móviles y para poder garantizar la escalabilidad de la red en términos de complejidad y de recursos necesarios (computacionales y capacidad de los enlaces).

Además, se busca explorar el uso de soluciones basadas en aprendizaje máquina para este tipo de problemas y generar las simulaciones necesarias de escenarios descentralizados que permitan obtener los datos requeridos para el entrenamiento y evaluación de los diferentes modelos.

Para alcanzar dichos objetivos, se han fijado los siguientes hitos:

1. Definir los elementos clave en los escenarios simulados para garantizar una representación válida y realista del entorno de comunicación.
2. Determinar la viabilidad de los modelos de DCB en la tarea de clasificación considerada, teniendo en cuenta las características específicas de los escenarios.
3. Implementar métodos adicionales de asignación para llevar a cabo una comparación de rendimiento con el modelo propuesto.
4. Analizar la eficiencia computacional del modelo para asegurar su viabilidad en términos de recursos y escalabilidad.
5. Desarrollar visualizaciones efectivas que permitan interpretar y comunicar los resultados de manera clara y accesible.

mientras se minimizan interferencias y garantizan requisitos de calidad de servicio.

[2] Los acrónimos hacen referencia a sus denominaciones en inglés, *Access Points* y *User Equipments* respectivamente.

[3] Referido a la eficiencia espectral, es decir, la velocidad de transmisión que se puede alcanzar para todos los UEs por unidad de ancho de banda disponible en el canal.

1.2 Motivación

La motivación detrás de este proyecto surge de la creciente necesidad de garantizar una conectividad de alta calidad independientemente de la ubicación de los UEs. En los sistemas inalámbricos tradicionales, la degradación de la señal en la periferia de las celdas limita el rendimiento y la eficiencia de la red, generando desigualdades en la calidad del servicio. Sin embargo, hoy en día, la mayor parte de aplicaciones que se usan y que triunfan entre los UEs requieren una conexión de alta calidad y homogénea independientemente de la ubicación, del entorno y de los patrones dinámicos de mobilidad de los UEs (servicios de *streaming*, redes sociales, navegación, *self-driving*, realidad aumentada, ...). Este proyecto aborda uno de los desafíos clave en la evolución de las redes inalámbricas: la asignación eficiente de APs en entornos descentralizados; un paso fundamental hacia la transformación de las infraestructuras de comunicación que en el futuro deben ser capaces de dar soporte a todas estas aplicaciones y a futuros casos de uso.

Además, este trabajo explora el uso de la inteligencia artificial y el aprendizaje automático como herramientas clave para abordar estos desafíos, proporcionando soluciones escalables y eficientes, capaces de adaptarse a escenarios dinámicos. De este modo, el proyecto no solo busca contribuir a la evolución de las redes inalámbricas avanzadas, sino que también introduce estrategias basadas en DCB y otros modelos de aprendizaje, promoviendo una asignación de recursos más inteligente y autónoma.

Personalmente, este proyecto me resulta muy atractivo porque se trata de un área de investigación muy presente en el día a día, donde los sistemas de comunicaciones inalámbricas son cada vez más esenciales. Poder trabajar en un ámbito donde los datos tienen una aplicación directa y práctica en problemas reales permite conectar el desarrollo teórico con necesidades concretas. Además, el estudio de aspectos relacionados con señales, electromagnetismo y transmisión de información me resulta particularmente atractivo, ya que me permite participar de forma activa en el avance de un campo que está en constante evolución. A todo ello se suma el interés por desarrollar estrategias de optimización y aplicar herramientas de aprendizaje automático en entornos complejos y dinámicos, lo que aporta un desafío adicional, haciendo el proyecto todavía más motivador.

1.3 Estructura

La memoria de este proyecto se estructura en los siguientes capítulos:

- **Capítulo 1:** El presente capítulo introduce los objetivos del proyecto, la motivación para llevarlo a cabo y la metodología considerada para acometer el plan de trabajo.
- **Capítulo 2:** Estado del arte y fundamentos teóricos. Este capítulo consistirá en una descripción de los conceptos teóricos y técnicas aplicadas en el proyecto.
- **Capítulo 3:** Recursos y herramientas. En él, se presentan los recursos hardware necesarios y las herramientas utilizadas durante el proyecto.
- **Capítulo 4:** Desarrollo de la solución. Se describen los procesos de diseño e implementación del modelo DCB junto con las aproximaciones adicionales.
- **Capítulo 5:** Resultados. Incluye una descripción de los escenarios de simulación utilizados, su proceso de desarrollo, y un análisis detallado de los resultados obtenidos a partir de las distintas aproximaciones.
- **Capítulo 6:** Conclusiones y trabajo futuro. Se recogen las principales conclusiones del proyecto, sus aportaciones, y se plantean posibles líneas de mejora o continuidad del trabajo desarrollado.

1.4 Metodología

En el desarrollo de software, una metodología de trabajo es un enfoque estructurado que guía la planificación, implementación y entrega de un proyecto. Define los procesos, la organización del trabajo, los roles involucrados y las herramientas utilizadas para gestionar el proyecto de manera eficiente. Su aplicación permite minimizar riesgos, optimizar la calidad del software y, en consecuencia, garantizar la satisfacción del UE final [6].

En este proyecto se ha adoptado una metodología iterativa incremental, un enfoque que consiste en desarrollar el software en pequeños incrementos y mejorarlo progresivamente a través de ciclos iterativos. Este método equilibra flexibilidad y mejora continua, permitiendo incorporar el *feedback* recibido y adaptarse tanto a cambios en los requisitos como a desafíos inesperados. Dado que tanto las tecnologías empleadas como la cuestión investigada en este trabajo están sujetas a una evolución constante y rápida innovación, esta metodología garantiza que el modelo permanezca actualizado y relevante durante su desarrollo. Además, reduce significativamente el riesgo asociado a grandes cambios, facilitando la implementación progresiva y la evaluación continua de los resultados. Por último, permite detectar (y corregir) potenciales errores en fases tempranas del desarrollo, minimizando su impacto final en el proyecto y el coste asociado.

El uso de un tablero Kanban como apoyo permite además un desarrollo más controlado y adaptable, asegurando que cada incremento del proyecto se implemente y evalúe de manera efectiva. Esta herramienta proporciona una visión clara del estado del proyecto, facilita la distribución y reorganización de tareas en tiempo real, permite realizar retrospectivas y evaluaciones periódicas, e identificar tareas atascadas. En la Figura 1.1, se muestra el tablero Kanban generado en Jira [7] y utilizado en este proyecto, que cuenta con 5 columnas: Backlog general, Backlog iteración, En curso, Finalizado iteración y Finalizado histórico. Esta organización permite visualizar tanto el estado global del proyecto como el progreso específico de la iteración actual, asegurando un flujo de trabajo eficiente.

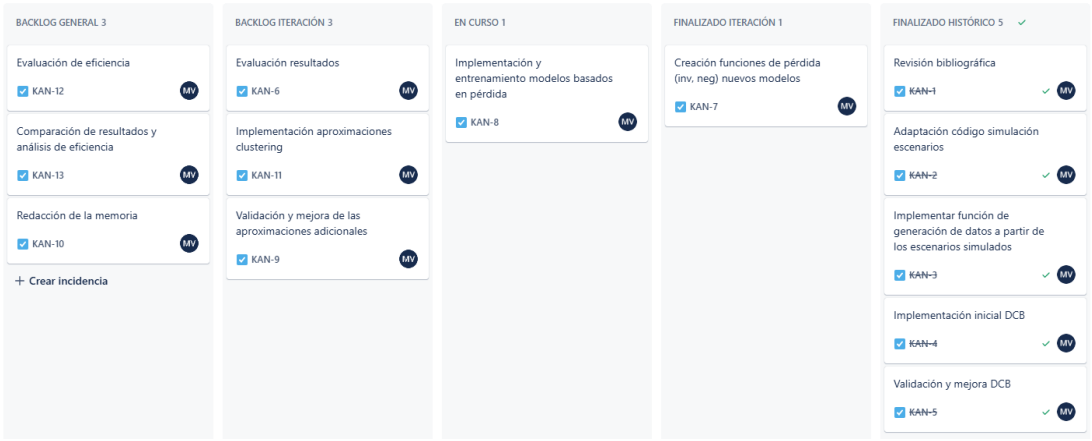


Figura 1.1: Tablero Kanban del proyecto. *(Generación propia)*

1.5 Planificación y costes

En esta sección se describe la planificación del proyecto, detallando las iteraciones en las que se organizó su desarrollo, así como la estimación de los costes asociados.

1.5.1 Planificación

La planificación del proyecto para la asignación de APs a UEs comenzó con una reunión inicial donde se planteó el escenario de comunicación considerado, el problema de optimización a resolver y se exploraron los modelos candidatos para abordarlo. Una vez decidido el uso de DCB, se estipularon las 7 iteraciones de trabajo detalladas a continuación:

- **Iteración 1 - Revisión bibliográfica inicial:** investigación exhaustiva sobre *Cell-Free Massive MIMO*, haciendo hincapié en el problema de *scheduling* y, más concretamente, en la asignación de APs considerando una visión *user-centric*. Revisión de la docu-

mentación sobre DCB y sus aplicaciones previas en el ámbito de las comunicaciones inalámbricas. En esta parte también se realizó el anteproyecto.

- **Iteración 2 - Desarrollo del modelo de simulación de escenarios:** revisión y adaptación del código de simulación de escenarios proporcionado en [2] para generar datos de entrada realistas. Esta fase es fundamental para definir los parámetros del entorno y garantizar que los escenarios reflejan condiciones representativas del problema.
- **Iteración 3 - Implementación inicial del modelo DCB:** determinación de la arquitectura del modelo, definiendo los estados, la formulación de la recompensa basada en el *sum-rate* y los mecanismos de interacción del agente. Entrenamiento inicial del modelo, y ajustes de hiperparámetros para optimizar su desempeño.
- **Iteración 4 - Implementación de aproximaciones adicionales:** desarrollo de modelos complementarios basados en *clustering* y funciones de pérdida personalizadas. Esta iteración permite explorar estrategias alternativas de asignación de APs.
- **Iteración 5 - Evaluación de eficiencia de los modelos:** análisis detallado del rendimiento computacional de cada modelo, considerando tiempos de ejecución, consumo de memoria y escalabilidad. Este análisis es clave para determinar la viabilidad práctica de cada aproximación en escenarios con alta densidad de UEs y APs.
- **Iteración 6 - Comparación de los modelos:** comparación entre el modelo DCB, una asignación aleatoria de APs y las estrategias adicionales desarrolladas. Esta evaluación se basa en métricas como el *sum-rate* medio y la capacidad de adaptación a diferentes configuraciones de red.
- **Iteración 7 - Redacción de la memoria:** redacción de la memoria del proyecto, documentando el proceso de desarrollo, implementación y evaluación de los modelos. En esta fase, se organizan los resultados obtenidos y se formulan las conclusiones del estudio.

Las actividades correspondientes a cada una de las iteraciones se representaron en un diagrama de Gantt creado, de nuevo, en Jira [7], y mostrado en la Figura 1.2. Este diagrama proporciona un cronograma visual para hacer seguimiento de las tareas y los hitos a lo largo del ciclo de vida del proyecto. Esta herramienta permite gestionar el tiempo de manera eficiente al exponer gráficamente la duración prevista de cada actividad y sus relaciones temporales, facilitando así la planificación y asegurando la finalización puntual de las tareas.

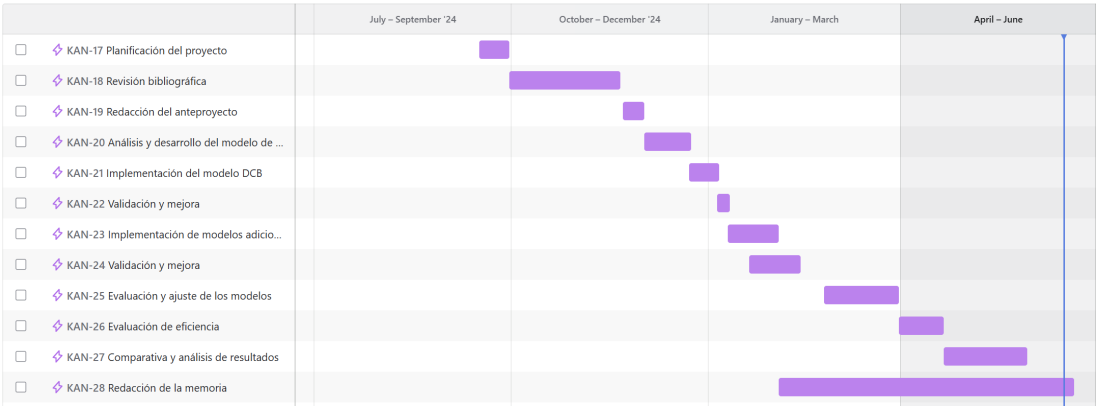


Figura 1.2: Diagrama de Gantt del proyecto. *(Generación propia)*

A lo largo del desarrollo del proyecto fue necesario realizar algunos ajustes y adaptaciones. Algunas fases, como la inicial de revisión bibliográfica, se extendieron algo más de lo previsto debido a diversos motivos como el solapamiento con las clases y el trabajo. El desarrollo de las simulaciones y del modelo DCB se completaron conforme a los tiempos estimados. Pero, durante estas etapas, fueron surgiendo nuevas ideas que llevaron a ampliar el alcance inicial. Entre estas, destaca la incorporación de dos variantes de modelos basados en funciones de pérdida, y la propuesta de un modelo adicional basado en *clustering* difuso como enfoque no supervisado. La integración de estos nuevos modelos requirieron reorganizar la planificación y extender los plazos inicialmente previstos, aunque se integraron sin generar grandes problemas de seguimiento. Durante el desarrollo del modelo de *clustering* se presentó un pequeño contratiempo relacionado con el ajuste del *threshold*, que se resolvió mediante la aplicación de un algoritmo de búsqueda por bisección para ajustar el parámetro óptimo en cada escenario. Finalmente, la redacción de la memoria se inició de forma solapada y fue avanzando progresivamente a medida que se completaban los distintos bloques de trabajo.

Asimismo, se llevaron a cabo diferentes reuniones con los tutores, con el objetivo de presentar avances, resolver dudas, recibir orientación y explorar nuevas ideas surgidas durante el proceso.

1.5.2 Costes

El presupuesto del proyecto está relacionado con los recursos asignados a cada una de las actividades desarrolladas. Para este proyecto, los recursos personales involucrados incluyen una ingeniera de datos (la estudiante) y dos tutores académicos que desarrollaron el rol de supervisores o directores del trabajo. Cada uno de estos recursos cuenta con un coste asociado por hora, lo que permite calcular los costes en función del tiempo dedicado a cada actividad.

En cuanto a los recursos computacionales, la ingeniera de datos cuenta con un ordenador personal con capacidad de cómputo suficiente desde el cual se desarrolló el proyecto, evitando la necesidad de adquirir hardware específico o hacer uso de plataformas adicionales para la ejecución de los experimentos.

En cuanto al software utilizado durante el desarrollo, se emplearon principalmente herramientas y librerías de código abierto, como *Python* junto con bibliotecas especializadas de procesamiento numérico y aprendizaje automático (*NumPy*, *Pandas*, *TensorFlow*,etc.). Todas ellas son de libre acceso y no han supuesto costes adicionales en licencias. Además, no fue necesario recurrir a ningún software de pago ni a servicios externos, por lo que el coste asociado al uso de software ha sido nulo.

La Tabla 1.1 presenta un desglose detallado del coste total estimado del proyecto, considerando las horas de trabajo dedicadas y los salarios correspondientes a cada recurso humano, obtenidos de [8]. Esta estimación facilita el seguimiento de los costes derivados de la realización del proyecto y contribuye a una gestión eficiente del presupuesto.

Recurso	**Coste/h**	**Tiempo dedicado**	**Coste total**
Ingeniera de datos *Junior*	14.10 €/h	300 h	4,230 €
Directores de proyecto	38.46 €/h	25 h	961.50 €
Total		325 h	5,191.50 €

Tabla 1.1: Tabla de costes estimados.

Capítulo 2

Estado del arte y fundamentos teóricos

Este capítulo tiene como finalidad proporcionar el contexto teórico y técnico necesario para comprender el problema abordado en el proyecto y justificar la elección de la metodología utilizada. En él, se presentarán los fundamentos conceptuales y el estado del arte en relación con los sistemas *Cell-Free Massive* *Multiple-Input Multiple-Output (MIMO)*, el problema de *scheduling* en redes inalámbricas y las técnicas de aprendizaje automático aplicadas a la asignación de punto de accesos (APs). Se revisarán los enfoques tradicionales, sus limitaciones y la motivación para la aplicación de Deep Contextual Bandits (DCB) en este problema. Se analizarán además enfoques alternativos, como el *clustering* difuso y modelos basados en pérdida, y se discutirán trabajos previos relevantes en este ámbito.

2.1 Sistemas celulares tradicionales

Desde su aparición en la década de 1980, los sistemas celulares han revolucionado las comunicaciones móviles, permitiendo la transmisión inalámbrica de voz y datos a través de extensas superficies geográficas. Estos sistemas se basan en la división del territorio en celdas, cada una equipada con una estación base (BS) que gestiona las comunicaciones dentro de su perímetro. Esta configuración facilita la reutilización de frecuencias en celdas no adyacentes, maximizando el uso del espectro disponible y aumentando la capacidad de la red (número de UEs servidos de forma simultánea). Por lo tanto, en esta arquitectura de red, cada UE está conectado a una única BS en cada instante de tiempo.

La evolución de las redes celulares se ha clasificado en generaciones, desde la 1G hasta la actual 5G, cada una introduciendo mejoras significativas en términos de capacidad, velocidad de transmisión y servicios ofrecidos. Como se visualiza en la Figura 2.1, a lo largo de estas

generaciones, se han ido introduciendo múltiples mejoras en los esquemas de modulación y codificación empleados, el acceso múltiple y la gestión de recursos, con el objetivo de mejorar la eficiencia y reducir las interferencias intra e intercelulares [9].

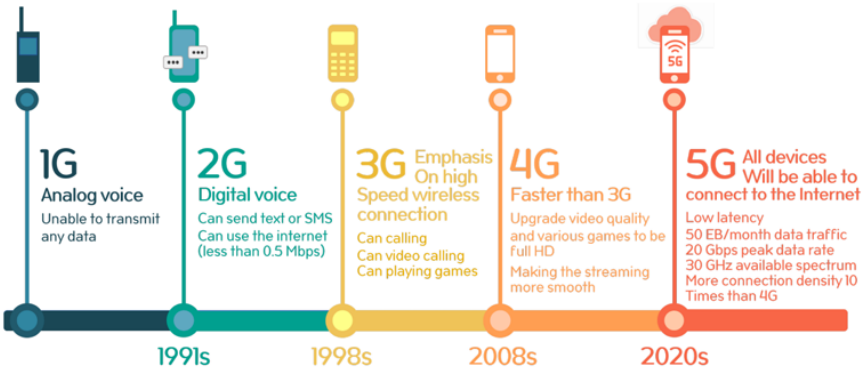


Figura 2.1: Evolución de las redes celulares [1].

A pesar de los avances, los sistemas celulares tradicionales se enfrentan a desafíos derivados de su arquitectura. Uno de los principales obstáculos es la degradación del rendimiento en los bordes de celda, cuando el UE está más alejado de la BS, ya que la interferencia es más pronunciada y la calidad del servicio (QoS) se reduce significativamente, lo que afecta negativamente la experiencia del UE. Para mitigar este problema, se han implementado múltiples técnicas relacionadas con el control de potencia o esquemas de codificación adaptativos. Sin embargo, estas soluciones presentan limitaciones en entornos urbanos densos o áreas con alta movilidad, donde la gestión de recursos y la coordinación entre celdas se vuelve más compleja. En ellos, las redes basadas en celdas fijas y gestión estática de los recursos, afrontan dificultades para satisfacer la creciente demanda de datos y la necesidad de conexiones más homogénas y dinámicas [9, 10].

2.2 Cell-Free Massive MIMO

Las redes *Cell-Free Massive MIMO* surgen como alternativa a las limitaciones de las estructuras celulares convencionales. A diferencia de los sistemas tradicionales basados en celdas, estas eliminan la dependencia de estaciones base centralizadas y distribuyen las antenas de acceso de manera uniforme a lo largo del área de cobertura, adoptando así una asignación más flexible y eficiente de los APs. En este tipo de sistemas, cada UE puede estar servido por múltiples BSs o APs de forma simultánea y que cooperan para mejorar la QoS. Como se menciona en [11], de este modo, se promete mejorar la cobertura, aumentar la capacidad y ofrecer una experiencia de UE más consistente, independientemente de la ubicación geográfica.

2.2.1 Arquitectura

En un sistema *Cell-Free Massive MIMO*, los APs se despliegan de manera distribuida por todo el área de cobertura, formando una red sin fronteras fijas de celdas. En lugar de que cada UE esté conectado exclusivamente a una estación base, como en los sistemas celulares tradicionales, ahora cada UE es servido simultáneamente por múltiples APs que cooperan a la hora de transmitir y recibir la información de cada UE. Además, a diferencia de los sistemas celulares donde cada estación base se compone de un número muy grande de antenas, en los sistemas *Cell-Free Massive MIMO*, tenemos un número grande de APs equipados con unas pocas antenas en cada uno de ellos. Esto permite mejorar la eficiencia espectral y reducir los problemas de interferencia y degradación de señal en las zonas de las celdas más alejadas respecto de la BS [12].

En la Figura 2.2 se muestran los dos tipos de sistemas celulares mencionados. El sistema celular, (a), y el *Cell-Free Massive MIMO*, (b).

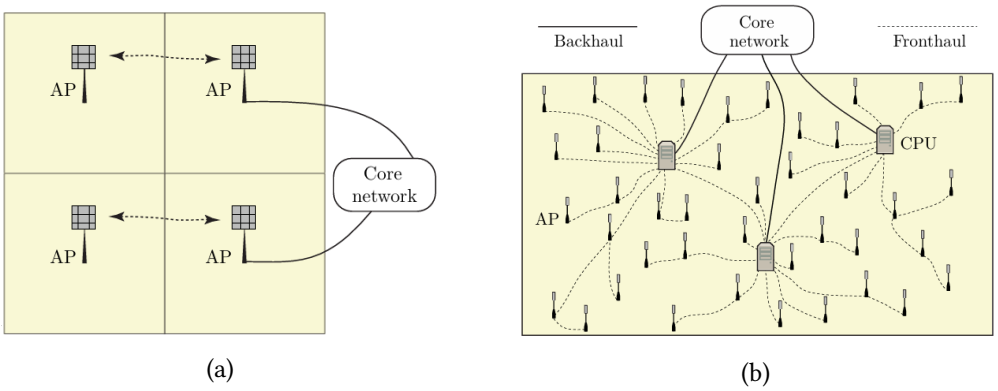


Figura 2.2: Sistemas de comunicación inalámbrica. (a) *Sistema celular con cuatro APs conectados a la red central.* (b) *Sistema Cell-Free con múltiples APs distribuidos por el espacio y conectados a los nodos CPUs, que a su vez están vinculadas con la red central* [2].

Además de los punto de accesos (APs) y los dispositivo usuarios (UEs), el sistema cuenta con un tercer elemento clave. Se dispone de una o varias Unidades de procesamiento centralizado (CPUs) que gestionan la asignación de recursos, el procesamiento de señales y la coordinación entre los APs.

2.2.2 Funcionamiento

El principio de funcionamiento en los sistemas *Cell-Free Massive MIMO* se basa en la cooperación dinámica entre múltiples APs para mejorar la QoS percibida por cada UE. Al

eliminar el concepto estático de celda, los UEs pueden establecer conexiones simultáneas con varios APs, pasando de este modo a un modelo más parecido al de un clúster o sub-grupo de APs que proporcionan servicio a cada UE. Además, como se puede apreciar en la Figura 2.2, la distribución eficiente de APs por todo el área de cobertura evita los problemas mencionados anteriormente relacionados con el hecho de que la QoS dependa de la ubicación y distribución de los UEs (que además puede cambiar de forma dinámica).

Este modelo permite también una mejor distribución de recursos en escenarios con alta densidad de UEs, mitigando la interferencia intercelular, un problema común en las redes celulares tradicionales. La colaboración entre APs, gestionada por la CPU central, permite una asignación dinámica y eficiente, garantizando una conectividad más estable para todos los UEs de la red.

2.3 *Scheduling* en redes inalámbricas

El *scheduling* en redes inalámbricas es un problema fundamental en la gestión de recursos. Se centra en optimizar la asignación de recursos (tiempo, frecuencia, potencia y espacio) entre múltiples UEs con el fin de maximizar la eficiencia espectral y reducir las posibles interferencias, optimizando así el rendimiento de la red y garantizando la QoS. A diferencia de las redes cableadas, donde los enlaces físicos están predefinidos, en los sistemas inalámbricos, donde los recursos son compartidos, una asignación ineficiente de los mismos puede generar desigualdad en el acceso, disminución del rendimiento global y una mayor latencia en la comunicación [13].

2.3.1 *Scheduling* en *Cell-Free Massive MIMO*

En redes *Cell-Free Massive MIMO*, el problema de *scheduling* se vuelve aún más complejo debido a la naturaleza distribuida de los APs y la falta de fronteras entre celdas. A diferencia de las redes celulares, donde cada UE está vinculado a una BS, en *Cell-Free Massive MIMO*, cada UE puede ser servido por múltiples APs de forma simultánea, lo que introduce nuevos retos en la coordinación y asignación de recursos.

Entre estos desafíos adicionales, se encuentran la necesidad de coordinar los APs para evitar interferencias y garantizar una procesamiento eficiente de la señal, de gestionar dinámicamente a los UEs debido a la movilidad de los dispositivos - los APs deben actualizar constantemente la asignación de recursos - y de maximizar el *sum-rate* global sin perjudicar a UEs con mala QoS. Como se propone en [11, 14], este objetivo puede expresarse formalmente como:

$$\max_{\{\mathcal{M}_k\}} \sum_{k=1}^{K} R_k,$$

donde:

- $\mathcal{M}_k$ representa el conjunto de APs asignados al UE k,
- R_k es la tasa de transmisión (*rate*) alcanzada por dicho UE. Esta tasa se suele medir en bits/s/Hz y representa realmente la eficiencia espectral, es decir, la capacidad de transmistir información para un determinado ancho de banda disponible [1].

Como se puede apreciar en la ecuación anterior, este problema de *scheduling* implica encontrar para cada UE el conjunto de APs que permiten maximizar el sum-rate global en el sistema, es decir, alcanzar la máxima velocidad de transmisión considerando todos los UEs activos y las posibles interfencias que ocasionan. Sin embargo, se trata de un problema cuya solución óptima se obtiene de forma combinatoria y, cuya complejidad computacional, crece dramáticamente a medida que se aumenta el número de APs y UEs. Además, la función de coste del problema y las restricciones asociadas no presentan una formulación convexa, lo que hace que se trate de un problema difícil de abordar mediante técnicas clásicas de optimización.

2.4 Aprendizaje por refuerzo

A diferencia del aprendizaje supervisado, donde el modelo aprende a partir de ejemplos etiquetados, o del aprendizaje no supervisado, que se enfoca en descubrir estructuras ocultas en los datos, el aprendizaje por refuerzo (Reinforcement Learning (RL)) se caracteriza por la toma de decisiones autónomas a través de agentes inteligentes[2]. Se puede apreciar en la Figura 2.3 una representación de estos 3 tipos de aprendizaje.

[1] En cualquier caso, existe una relación lineal directa entre eficiencia espectral y velocidad de transmisión, por lo que se pueden emplear estos dos términos prácticamente como sinónimos

[2] Los agentes autónomos son sistemas capaces de actuar de manera independiente en función del contexto en el que se encuentran.

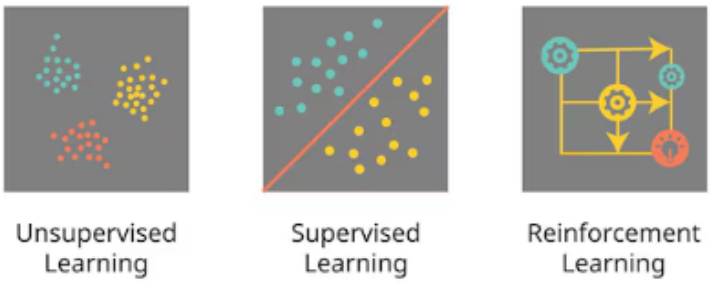


Figura 2.3: Representación gráfica de los 3 tipos de enfoques de aprendizaje automático mencionados [3].

El aprendizaje por refuerzo (RL) se basa en la interacción entre un agente y su entorno, con el objetivo de maximizar una señal de recompensa acumulada a lo largo del tiempo a través de realizar una serie de acciones seleccionadas de un amplio conjunto de posiles acciones. A diferencia de otros enfoques, aquí el agente aprende mediante prueba y error, sin necesidad de supervisión ni guías predefinidas por humanos. Esto le permite tomar decisiones secuenciales en escenarios no deterministas, adaptándose dinámicamente a la variabilidad del entorno al optimizar sus acciones en función del *feedback* recibido [15, 16].

2.4.1 Funcionamiento

En términos generales, un problema de RL se modela como un proceso de decisión de Markov (MDP), donde el agente observa un estado del entorno, elige una acción según una determinada política (*policy*), y recibe una recompensa y un nuevo estado en función de la dinámica del entorno [15]. Formalmente, un MDP se define por la tupla (S,A,P,R,γ), donde:

- $\boldsymbol{S}$: conjunto de estados.
- $\boldsymbol{A}$: conjunto de acciones disponibles.
- $\boldsymbol{P(s'|s,a)}$: probabilidad de que, tomando la acción a, se transicione del estado actual al estado s'.
- $\boldsymbol{R(s,a)}$: recompensa esperada al ejecutar la acción a en el estado s.
- γ: factor de descuento. Pondera la importancia de las recompensas futuras.

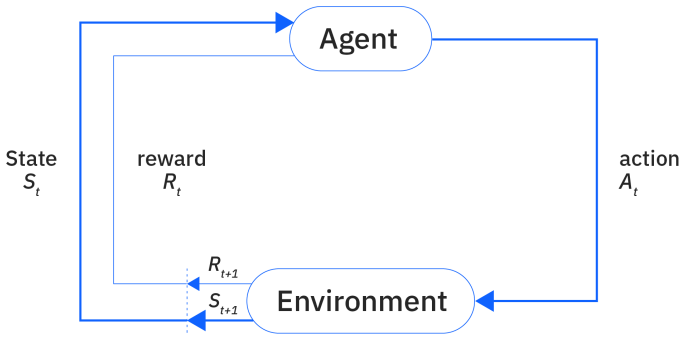


Figura 2.4: Representación de un escenario típico de aprendizaje por refuerzo [4].

La Figura 2.4 muestra una representación gráfica del funcionamiento de un modelo de RL, donde un agente lleva a cabo acciones en un entorno, este es interpretado, convertido en una representación estado-recompensa y enviado de vuelta al agente.

Al aprender mediante prueba y eror, uno de los retos fundamentales a los que se enfrente el RL es el equilibrio entre exploración y explotación. La primera implica seleccionar acciones que maximicen la recompensa esperada según el conocimiento actual del agente, mientras que la segunda requiere probar acciones alternativas para descubrir mejores estrategias a largo plazo.

2.5 *Contextual Bandits*

Dentro del aprendizaje por refuerzo, los Contextual Bandits (CB) representan un caso simplificado pero poderoso. Mientras que, en el problema clásico de *multi-armed bandits*, el agente debe elegir entre varias acciones para maximizar su recompensa en un entorno estacionario, los *contextual bandits* introducen un elemento adicional: el contexto. En cada iteración, el agente recibe información contextual que describe el estado actual del entorno. Basándose en este contexto, el agente selecciona la acción que considera más adecuada [17].

A diferencia de los proceso de decisión de Markov, los Contextual Bandits no consideran una dependencia entre estados pasados y futuros. Cada decisión es independiente y se basa únicamente en el contexto actual, lo que los hace útiles en aplicaciones donde las decisiones deben adaptarse a características específicas del entorno, como es la asignación de recursos en redes inalámbricas.

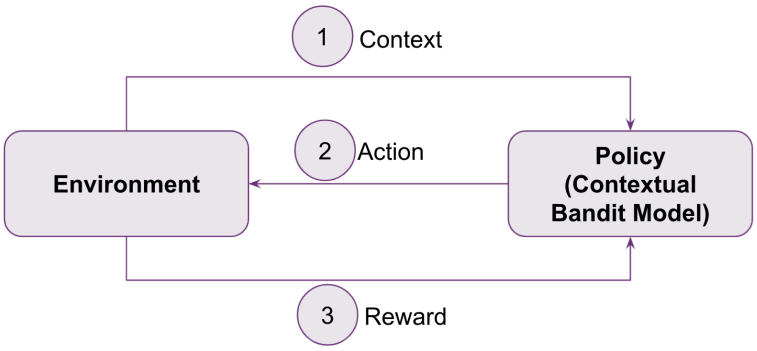


Figura 2.5: Representación de un escenario típico de *bandits* contextuales [5]

.

Como se aprecia en la Figura 2.5, en el caso de los CB, el agente lleva a cabo acciones en un entorno, este es interpretado, convertido en una recompensa y mandado, esta vez junto con el contexto del entorno, de vuelta al agente.

Si bien los CB tradicionales utilizan modelos lineales para mapear contextos a recompensas, estas aproximaciones pueden ser insuficientes cuando la relación entre el contexto y la recompensa es no lineal. Es por ello que surgen los Deep Contextual Bandits (DCB), que combinan la capacidad de las redes neuronales profundas para capturar patrones complejos con la exploración estructurada de los *bandits* [18].

2.6 *Deep Contextual Bandits*

Los DCB utilizan redes neuronales para aprender representaciones del contexto y predecir las recompensas asociadas a cada acción posible. Además, emplean técnicas como el muestreo de Thompson[3] (*Thompson Sampling*), que equilibra la exploración y la explotación al seleccionar acciones basadas en la probabilidad de ser óptimas según las creencias o conocimiento actual del modelo [19].

2.6.1 Funcionamiento

Su funcionamiento puede descomponerse en los siguientes pasos principales:

1. **Observación del contexto:** En cada interacción, el agente observa un *vector de características* que representa el estado actual del entorno. Este contexto puede incluir

[3] *Thompson Sampling* es un algoritmo de toma de decisiones donde las acciones se eligen de forma secuencial, equilibrando la explotación de la información conocida que maximiza el rendimiento inmediato con la exploración de nuevas acciones que puedan mejorar el rendimiento futuro.

información sobre el UE, condiciones del sistema o cualquier otro dato relevante para la toma de decisiones.

2. **Predicción de recompensas:** El contexto se introduce en una red neuronal que ha sido entrenada para estimar la recompensa esperada asociada a cada acción posible. Esta red actúa como aproximador no lineal de la función de valor o recompensa.

3. **Incorporación de incertidumbre:** Para no limitarse siempre a las acciones conocidas, el agente introduce cierto grado de variabilidad en sus predicciones. Esta variación permite que, en lugar de elegir siempre la opción aparentemente mejor, explore otras alternativas que podrían resultar más beneficiosas.

4. **Selección de la acción (*Thompson Sampling*):** Se genera una predicción estocástica de recompensa para cada acción, y se selecciona aquella con mayor valor muestreado. Siguiendo el planteamiento del algoritmo de *Thompson Sampling* para *bandits* contextuales [20], esto se representa como:

$$a_t = \arg\max_{a \in \mathcal{A}} \tilde{r}_t(a),$$

donde:

- $\tilde{r}_t(a)$ es una muestra de la distribución de probabilidad sobre la recompensa esperada de cada acción a en el instante t.

Esta muestra se genera en base a las creencias actuales del modelo, lo que permite balancear la explotación de acciones con alto valor esperado y la exploración de alternativas menos conocidas.

5. **Actualización del modelo:** Tras ejecutar la acción seleccionada, el agente observa la recompensa real obtenida. Este dato se incorpora al conjunto de entrenamiento y la red neuronal se actualiza, mejorando progresivamente su capacidad predictiva.

2.6.2 Aplicación de los *DCB* en la asignación de *APs*

El problema de asignar dinámicamente APs (APs) a dispositivos de UE (UEs) en sistemas *Cell-Free Massive MIMO* plantea múltiples desafíos: la naturaleza altamente distribuida del entorno, la variabilidad del canal y la necesidad de adaptarse continuamente a condiciones cambiantes. Ante este panorama, los modelos de DCB se presentan como una opción especialmente prometedora debido a características clave mencionadas anteriormente:

- Toman decisiones en función de un contexto dinámico, como las condiciones de canal y la interferencia entre UEs.

- Permiten actualizar el modelo continuamente conforme se recopilan nuevas observaciones.
- Encuentran mejores combinaciones de APs a largo plazo sin estancarse en soluciones subóptimas.
- Capturan relaciones no lineales complejas entre las características del entorno y las recompensas esperadas.

En este proyecto, el problema se aborda como una tarea de toma de decisiones contextualizada, donde se busca aprender una política que asigne dinámicamente los APs a cada UE en función de las condiciones del entorno. En esta situación, el modelo DCB, al operar en escenarios con alta variabilidad, ofrece un enfoque eficaz para identificar combinaciones de APs que maximizan el rendimiento del sistema. Además, es importante destacar que la asignación de APs se optimiza de forma específica para unas condiciones del entorno (ubicación y distribución de UEs, condiciones de canal, interferencias, ...) en un determinado instante de tiempo. Los valores de estas características no dependen necesariamente de otros instantes de tiempo (es decir, no hay correlación temporal), por lo que una solución basada en DCB es preferible respecto a otras aproximaciones de RL.

Previamente, para cada UE k, puede definirse su tasa de transmisión individual como:

$$R_k = \log_2 \left(1 + \sum_{i \in \mathcal{A}_k} \frac{\beta_{i,k}}{I_{i,k} + \sigma^2} \right), \tag{2.1}$$

donde:

- $\mathcal{A}_k$ es el conjunto de APs asignados al UE k,
- $\beta_{i,k}$ es la ganancia de canal entre el AP i y el UE k,
- $I_{i,k}$ es la interferencia entre el AP i y el UE k,
- N_0 es la potencia de ruido.

Cada acción es evaluada mediante una función de recompensa basada en una versión modificada del *sum-rate*, que refleja tanto la calidad de la asignación como ciertos aspectos de regularización. Asumiendo que tenemos K UEs en el sistema y dado un conjunto de vectores binarios de acción $\mathbf{a}_k = [a_{k,1}, \ldots, a_{k,M_k}]$ para cada UE k, donde $a_{k,i} = 1$ si el AP i es activado para el UE k, la recompensa se calcula como la suma de las tasas individuales R_k, definidas anteriormente en (2.1), junto a términos de regularización adicionales:

$$rw = \sum_{k=1}^{K} R_k + \lambda_1 \cdot \textit{scaling\_factor} + \lambda_2 \cdot \text{penalización}_{\text{acción\_rep}}. \tag{2.2}$$

Esta formulación permite al modelo aprender asignaciones que maximizan la eficiencia espectral mientras evita soluciones subóptimas. Además, al no depender de una formulación analítica fija de la relación señal-interferencia-más-ruido (SINR)[4][21], el sistema puede adaptarse mejor a la información empírica obtenida en cada escenario simulado.

En definitiva, los DCB proporcionan un marco flexible, escalable y adaptativo para afrontar el problema de asignación dinámica de APs en sistemas *Cell-Free Massive MIMO*. Sus características los convierten en una alternativa especialmente adecuada frente a métodos más estáticos o estructurados, lo que justifica su elección como eje central de este trabajo.

2.7 Enfoques alternativos

Aunque los modelos de DCB representan una solución avanzada para la asignación dinámica de APs, en este proyecto también se han explorado enfoques alternativos con el objetivo de tener referencias adicionales a la hora de comparar los resultados obtenidos. Entre estos enfoques destacan el uso de técnicas de *clustering* difuso (*fuzzy clustering*) y modelos basados en la optimización directa de funciones de pérdida relacionadas con la eficiencia espectral.

Estos métodos también permiten desarrollar soluciones razonables al problema considerado en sistemas *Cell-Free Massive* MIMO, pero presentan ciertas limitaciones en su capacidad de adaptación dinámica al entorno. Sin embargo, su implementación permite establecer bases de comparación muy útiles para evaluar el rendimiento del enfoque propuesto basado en RL.

2.7.1 *Fuzzy Clustering*

El *fuzzy clustering* o *clustering difuso* es una técnica de agrupamiento en la que cada elemento puede pertenecer a varios grupos simultáneamente, con distintos grados de pertenencia. A diferencia del *clustering* tradicional, donde cada instancia se asigna a un único grupo, el enfoque difuso permite una representación más flexible y continua de la relación entre los datos y las clases [22, 23].

[4] En lugar de utilizar una expresión matemática detallada del SINR, basada en modelos físicos del canal, se emplea una formulación más simple construida a partir de los valores simulados de ganancia e interferencia. Esto permite una implementación más flexible y centrada en los datos disponibles.

En el contexto de este proyecto, se ha aplicado *fuzzy clustering* como un mecanismo para seleccionar los subconjuntos de APs que deben proporcionar servicio a cada UE. La idea principal consiste en calcular la similitud (en función de la distancia o de la ganancia de canal) entre los UEs y los APs, y utilizar dicha información para formar clústeres difusos donde cada AP puede contribuir en distinta medida a múltiples UEs.

Este enfoque tiene como objetivo reducir la interferencia y mejorar la cobertura sin recurrir a un modelo entrenado. Aunque no realiza un aprendizaje adaptativo, su implementación resulta sencilla y computacionalmente eficiente, lo que lo convierte en una alternativa razonable en entornos donde se prioriza la rapidez de decisión sobre la precisión óptima.

Funcionamiento

El algoritmo de *Fuzzy C-Means* (FCM), uno de los más utilizados y descrito originalmente en [22], parte de una inicialización aleatoria de los centros de clústeres. Luego, alterna entre dos pasos principales:

1. **Cálculo de grados de pertenencia:** Cada punto de datos x_k se asocia a cada clúster i con un valor $u_{ik} \in [0, 1]$ que se calcula como:

$$u_{ik} = \left(\sum_{j=1}^{C} \left(\frac{d(x_k, c_i)}{d(x_k, c_j)} \right)^{\frac{2}{m-1}} \right)^{-1}$$

   donde:

   - $d(x_k, c_i)$ es la distancia entre el dato y el centro del i-ésimo clúster,
   - $m > 1$ es el coeficiente de difusividad (cuanto mayor sea este, más se repartirá la pertenencia del dato entre los clusters),
   - C es el número total de clústeres.

2. **Actualización de los centros de los clústeres:** Los centros se recalculan como una media ponderada de todos los puntos de datos, según sus grados de pertenencia. Así, los puntos que pertenecen más fuertemente al clúster tienen mayor influencia en su posición:

$$c_i = \frac{\sum_{k=1}^{N} u_{ik}^m x_k}{\sum_{k=1}^{N} u_{ik}^m},$$

   donde, además de los elementos anteriores, tenemos que:

   - u_{ik} es el grado de pertenencia calculado en el paso previo,

- N es el número total de puntos de datos.

3. El proceso se repite hasta que los centros convergen o se alcanza un número máximo de iteraciones.

Ejemplo

Supongamos un escenario simple con tres APs y un único UE. En lugar de asignar el UE a un solo AP, el algoritmo de *fuzzy clustering* calcula un grado de pertenencia para cada uno. Por ejemplo:

- AP_1: $u_{0k} = 0.7$,
- AP_2: $u_{1k} = 0.4$,
- AP_3: $u_{2k} = 0.2$.

En este caso, el UE estaría más vinculado al AP_1, pero también se reconoce una relación más débil con el AP_2 y el AP_3.

Añadamos ahora otro UE y otros dos APs adicionales. Nos situamos de este modo en un escenario donde se pueden encontrar 2 UEs y 5 APs. En este caso, el algoritmo devolvería una matriz con la siguiente forma:

	AP_0	**AP_1**	**AP_2**	**AP_3**	**AP_4**
UE_0	0.7	0.4	0.2	0.5	0.9
UE_1	0.3	0.6	0.8	0.5	0.1

Tabla 2.1: Representación de la matriz de pertenencias.

Como se aprecia en la Tabla 2.1, las filas se corresponden con los UEs mientras que en las columnas se sitúan los APs. Cabe destacar que la suma de cada columna es 1. La pertenencia de un AP a todos los UEs será exactamente 1, de este modo se identifica claramente la superioridad de unos clústeres (UEs) frente a otros.

Visualmente, el resultado sería algo similar al diagrama mostrado en la imagen 2.6.

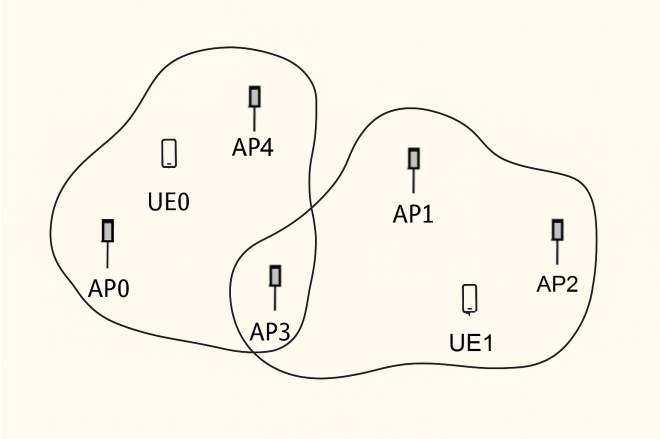


Figura 2.6: Clústeres de pertenencia de APs a UEs. *(Generación propia)*

Ventajas y limitaciones

Una de las principales ventajas del *fuzzy clustering* es la simplicidad tanto conceptual como computacional, que permite una implementación rápida sin necesidad de entrenamiento supervisado ni grandes cantidades de datos. Además, su estructura flexible permite asignaciones parciales, lo que es excelente para situaciones en las que las decisiones no deben ser exclusivamente binarias.

Sin embargo, este enfoque presenta limitaciones importantes en escenarios complejos o dinámicos. Al no incorporar un mecanismo de aprendizaje adaptativo, no puede ajustar sus decisiones en función de resultados previos ni anticipar configuraciones más óptimas. Esto lo hace menos preciso y menos robusto frente a las condiciones cambiantes del entorno en comparación con modelos de aprendizaje automático como los DCB. Asimismo, es un algoritmo muy dependiente del umbral, lo que en el problema de la asignación de APs es extremadamente importante, ya que es el factor que determina qué APs sirven a cada UE.

2.7.2 Modelos basados en pérdida

Los modelos basados en pérdida constituyen otro enfoque alternativo para resolver el problema de asignación de APs. En lugar de aprender una política de asignación a partir de interacciones con el entorno, estos modelos se entrenan para minimizar directamente una función de pérdida que refleja el objetivo deseado, como puede ser maximizar el *sum-rate* o minimizar su inverso; funciones inspiradas en formulaciones típicas de la optimización de recursos en comunicaciones inalámbricas [15, 24, 25].

Este tipo de modelos se enmarcan dentro del aprendizaje supervisado. En este proyecto, se

han explorado dos situaciones: la maximización del *sum-rate* y la minimización de su inverso. Las definiciones matemáticas para las funciones de optimización utilizadas se detallarán en capítulos posteriores; en esta sección, se describirá de forma general el funcionamiento.

Funcionamiento

Estos modelos comparten similitudes conceptuales con los modelos de aprendizaje supervisado tradicionales. Ambos aprenden ajustando sus parámetros para minimizar una función de error que compara la salida del modelo con un objetivo deseado. Sin embargo, los modelos basados en pérdida se diferencian en que su objetivo no es predecir etiquetas explícitas. En su lugar, se centran exclusivamente en minimizar una función de pérdida personalizada.

En el contexto de este proyecto, el funcionamiento de estos modelos sigue las siguientes etapas:

1. **Entrada del entorno:** El modelo recibe como entrada información contextual relevante, como las ganancias de canal y las interferencias entre UEs.
2. **Propuesta de asignación:** A partir de las entradas, el modelo genera una propuesta de asignación de APs a UEs.
3. **Evaluación mediante función de pérdida:** La asignación propuesta se evalúa calculando una recompensa basada en la eficiencia espectral (*sum-rate*) alcanzada, y se compara contra el objetivo deseado mediante una función de pérdida.
4. **Actualización del modelo:** A partir del valor obtenido para la pérdida, se actualizan los parámetros del modelo para mejorar sus futuras propuestas de asignación.

Este proceso permite que el modelo aprenda a generar asignaciones más eficientes sin necesidad de explorar dinámicamente el entorno ni recibir retroalimentación secuencial, lo que lo convierte en una alternativa muy atractiva para escenarios donde se dispone de información de entrenamiento estática o simulada, como es este caso.

Ventajas y limitaciones

Una de las principales ventajas de los modelos basados en pérdida es que permiten optimizar directamente la métrica deseada, como el *sum-rate*, sin necesidad de diseñar políticas de exploración o de gestionar la interacción continua con el entorno. Esto simplifica tanto el diseño como el proceso de entrenamiento, reduciendo la complejidad del modelo y los tiempos necesarios para su ajuste.

Sin embargo, presentan también numerosas limitaciones. Al no incorporar mecanismos de exploración ni retroalimentación secuencial, carecen de adaptabilidad dinámica frente a cambios en el entorno. Esto puede no afectar significativamente al proyecto actual, dado que las simulaciones empleadas presentan una naturaleza estática. No obstante, sería un factor a considerar en trabajos futuros que modelen escenarios de redes inalámbricas más dinámicos, donde las condiciones varíen en tiempo real. Además, su rendimiento puede verse limitado cuando se enfrentan a condiciones que difieren de las vistas durante el entrenamiento, ya que no disponen de estrategias de ajuste en tiempo real. Por último, este tipo de modelos son muy sensibles a los parámetros y configuraciones del sistema, ya que si estas cambian (por ejemplo, el número de APs o el número de UEs), también cambian las dimensiones del modelo y puede ser necesario reajustar el proceso de entrenamiento.

2.8 Trabajos relacionados

Diversos trabajos han abordado el problema de asignación de recursos en redes inalámbricas, con especial énfasis en sistemas *Cell-Free Massive MIMO*. En esta sección, se revisan brevemente algunas de las aproximaciones más relevantes, contextualizando el enfoque de este proyecto basado en DCB.

Los primeros enfoques aplicados a *Cell-Free Massive MIMO* se basaron en aplicar técnicas heurísticas clásicas, como el *round-robin scheduling*, el *greedy user selection* o métodos de asignación secuencial basada en la ganancia del canal más fuerte [24, 26]. Si bien estas estrategias permiten una asignación rápida y de baja complejidad computacional, presentan importantes limitaciones en términos de adaptabilidad y eficiencia espectral, especialmente en entornos dinámicos o de alta densidad de UEs.

El aprendizaje por refuerzo profundo (DRL) también se ha aplicado en la asignación de recursos en redes inalámbricas, empleando algoritmos como *Deep Q-Network* (DQN) o *Actor-Critic* para optimizar políticas de asignación basadas en la experiencia [27, 28]. Estos enfoques ofrecen gran capacidad de adaptación, pero tienen elevados requisitos computacionales y necesitan largos procesos de exploración para alcanzar un desempeño óptimo.

Más recientemente, se exploró el uso de redes neuronales basadas en grafos (GNNs) para abordar el problema de asignación de APs en entornos de comunicación *cell-free*. En [29], se propone un modelo basado en GNNs capaz de capturar de forma explícita las relaciones espaciales y de interferencia entre UEs y APs, ofreciendo mejoras en términos de eficiencia espectral respecto a métodos tradicionales. Sin embargo, estos modelos suelen requerir con-

juntos de datos extensos para su entrenamiento y podrían perder capacidad de generalización en escenarios dinámicos.

A pesar de su potencial, los métodos basados en *Contextual Bandits* han recibido escasa atención en el ámbito de las redes inalámbricas. Además, su capacidad para tomar decisiones rápidas en función de observaciones contextuales, sin modelar explícitamente la dinámica del entorno, los convierte en una alternativa especialmente atractiva para problemas de asignación de recursos. Esta falta de investigación previa con este tipo de aproximaciones refuerza la motivación del presente proyecto, que busca aplicar técnicas de DCB como estrategia central. En esta misma línea, se puede destacar que tampoco existen demasiados trabajos que exploren el uso de técnicas como *fuzzy clustering* o modelos basados en optimizar funciones de pérdida para resolver este tipo de problemas de asignación de recursos.

2.9 Notación

En esta sección, se presenta un resumen de la notación utilizada a lo largo de la memoria. En la Tabla 2.2, se recogen los principales parámetros, variables y funciones empleados en la formulación de los modelos y algoritmos descritos. En general, se usarán letras sin formato para referirnos a variables escalares, letras en negrita para hacer referencia a vectores y letras caligráficas cuando se trate de conjuntos o secuencias.

Tabla 2.2: Resumen de notación empleada a lo largo de la memoria.

Símbolo	**Descripción**
L	Número total de APs en el escenario
K	Número de dispositivos UE (UEs) en el escenario
M_i	Número de APs considerados por el UE i
N	Número de antenas disponibles para cada AP
$\mathcal{M}_k$	Conjunto con los índices de los APs que proporcionan servicio al UE k
τ_p	Número de pilotos ortogonales usados para estimación de canal

Tabla 2.2 – continuación

Símbolo	Descripción
τ_u	Número de símbolos dedicados a la transmisión en el enlace de subida
τ_c	Tamaño del intervalo de coherencia, es decir, número total de símbolos que se pueden transmitir con las mismas condiciones de canal
ASDφ^4	Desviación estándar (en radianes) del ángulo acimutal en el modelo de dispersión local
$\mathbf{a} = [a_1, \ldots, a_L]$	Vector binario de acción; $a_i = 1$ si el AP i está activo
R_k	Tasa de transmisión (rate) del UE k
SINR$_k$	Relación señal-interferencia-ruido para el UE k
$\beta_{i,k}$	Ganancia del canal del UE k en su enlace con el AP i
I_i	Interferencia causada por el resto de UEs sobre el AP i
N_0	Desnsidad espectral del ruido térmico
λ_1, λ_2	Coeficientes de regularización en la función de recompensa
scaling_factor	Factor de escalado que premia asignaciones equilibradas
$penalización_{acción\_rep}$	Penalización por repetir la misma acción en decisiones sucesivas
u_{ik}	Grado de pertenencia del punto de datos k al clúster i (*clustering* difuso)
c_i	Centro del clúster i
$d(x_k, c_i)$	Distancia entre el dato x_k y el centro del clúster c_i

m	Coeficiente de difusividad en Fuzzy C-Means

[4] Es una medida de cuánto se dispersa la energía en ángulos horizontales alrededor de un UE en el modelo de canal.

Capítulo 3

Recursos y herramientas

En este capítulo se presentan los recursos materiales y las herramientas software empleadas a lo largo del desarrollo del proyecto. En las siguientes secciones se describen tanto los aspectos relacionados con la infraestructura hardware disponible como los entornos de programación, bibliotecas y plataformas utilizadas para la implementación, entrenamiento y evaluación de los modelos desarrollados.

3.1 Recursos hardware

El desarrollo del proyecto requirió recursos computacionales capaces de soportar tanto la generación de datos a partir de escenarios simulados como el entrenamiento y evaluación de modelos de aprendizaje automático. Estas tareas implican operaciones matriciales intensas y el procesamiento de volúmenes considerables de datos, por lo que fue necesario contar con una tarjeta gráfica dedicada que permitiera acelerar los cálculos, así como una memoria RAM suficiente para almacenar y acceder a dichos datos durante las fases de entrenamiento.

Como equipo principal se empleó un ordenador portátil con procesador Intel, 16 GB de memoria RAM, unidad SSD NVMe con capacidad suficiente (1 TB) para almacenar los datos simulados, los resultados de entrenamiento y otros archivos generados durante el desarrollo; y una tarjeta gráfica dedicada NVIDIA GeForce RTX 4060 Laptop con 16 GB de memoria GDDR6. Este conjunto ofreció un rendimiento adecuado para entrenar redes neuronales, ejecutar simulaciones de escenarios y realizar análisis posteriores sin presentar cuellos de botella significativos. El equipo incluye además una GPU integrada (Intel Arc), que no fue utilizada en este proyecto [30].

También se utilizó una máquina virtual con sistema operativo Ubuntu como entorno de trabajo adicional. Su uso permitió no solo validar la portabilidad de los *scripts* a entornos de

tipo servidor, sino también facilitar el aprovechamiento de la GPU en tareas de entrenamiento mediante librerías como TensorFlow, gracias a su mejor compatibilidad con dicho sistema operativo.

A continuación, se resumen de forma esquemática los principales recursos empleados:

- **Procesador (CPU):** Intel Core Ultra 9.
- **Memoria RAM:** 16 GB.
- **Tarjeta gráfica (GPU):** NVIDIA GeForce RTX 4060 Laptop con 16 GB de memoria GDDR6.
- **Almacenamiento:** SSD NVMe.
- **Sistemas operativos:** Windows 11 como entorno principal de desarrollo, y Ubuntu en máquina virtual para entrenamiento y compatibilidad con bibliotecas de *deep learning*.

3.2 Herramientas de desarrollo

Además de los recursos hardware, se usaron diversas herramientas de software que facilitaron tanto el desarrollo de las simulaciones y los modelos como la gestión de los datos y la presentación de resultados. Estas herramientas fueron seleccionadas en función de su compatibilidad, su rendimiento y su facilidad de integración en entornos de trabajo orientados al aprendizaje automático.

El lenguaje de programación seleccionado fue Python, debido a su amplia disponibilidad de librerías orientadas a *machine learning* y su facilidad para el manejo de datos estructurados [31]. Para el entrenamiento de modelos, se emplearon *frameworks* de aprendizaje profundo compatibles con GPU que permitieron optimizar el rendimiento de los experimentos. Además, se utilizaron plataformas de desarrollo colaborativo y de gestión de versiones que facilitaron la organización del código y la reproducibilidad de los experimentos en distintos entornos.

A continuación, se describen los entornos de programación utilizados, las principales librerías empleadas para la implementación de los modelos, y herramientas complementarias que han apoyado el desarrollo global del proyecto.

3.2.1 Entornos de desarrollo

Todos los procesos, tanto la generación de escenarios simulados como el entrenamiento y validación de los modelos, se realizaron de manera local. Para gestionar de forma ordena-

da las dependencias y facilitar la portabilidad entre sistemas, se utilizaron entornos virtuales adaptados a cada sistema operativo.

En Windows, se creó un entorno virtual `venv` basado en Python 3.9. En Ubuntu, se configuró un entorno independiente utilizando Python 3.10.12 y TensorFlow 2.11.0, asegurando así la compatibilidad con la aceleración por GPU y aprovechando las capacidades nativas del sistema para aprendizaje automático. Esta estrategia permitió mantener la coherencia en la ejecución de los experimentos y facilitar la transición entre los distintos sistemas operativos empleados.

El desarrollo y la implementación se llevaron a cabo principalmente en *Visual Studio Code* (VS Code), un editor de código fuente ligero y multiplataforma ampliamente utilizado en entornos de ciencia de datos y aprendizaje automático, donde se consolidó como una de las plataformas más recomendadas [32]. Destaca por su rapidez, su extensibilidad a través de extensiones especializadas y su excelente integración con entornos virtuales y sistemas de control de versiones como Git, del que hablaremos un poco más adelante.

3.2.2 Librerías utilizadas

También se emplearon diversas librerías de Python orientadas al cálculo numérico, la manipulación de datos, la visualización de resultados y el entrenamiento de modelos de aprendizaje automático. A continuación, se describen brevemente las más relevantes en el contexto del proyecto:

- **NumPy**: es una librería fundamental para el procesamiento numérico en Python, proporcionando estructuras de datos eficientes como los arrays multidimensionales y funciones optimizadas para operaciones matemáticas y de álgebra lineal [33]. En este proyecto se utilizó como base para operaciones matriciales y generación de datos simulados.

- **Pandas**: es una librería de análisis y manipulación de datos que proporciona estructuras de alto nivel como *DataFrame* para trabajar de forma eficiente con datos tabulares [34]. En el proyecto facilitó la organización, limpieza y análisis de datos en formato tabular, así como el preprocesamiento y análisis de los resultados tanto de las simulaciones como de las evaluaciones.

- **SciPy**: complementa a NumPy ofreciendo algoritmos avanzados de álgebra lineal, optimización, estadísticas y procesamiento de señales [35]. El submódulo `scipy.linalg` se empleó en este trabajo para resolver sistemas de ecuaciones y realizar operaciones de descomposición de matrices.

- **Matplotlib**: es la principal librería de visualización en Python, permitiendo la creación de gráficos estáticos, animados e interactivos [36]. Se utilizó inicialmente para representar gráficamente la evolución de las métricas de rendimiento, como el sum-rate y el uso de memoria y coste computacional de los modelos durante el análisis de eficiencia de los mismos.

- **TensorFlow y Keras**: TensorFlow es una plataforma de código abierto para *machine learning* que permite la implementación de redes neuronales profundas optimizadas para ejecución en CPU y GPU [37]. Ambas herramientas fueron usadas para la construcción, entrenamiento y evaluación de los modelos de aprendizaje automático. Keras, como interfaz de alto nivel integrada en TensorFlow, simplificó la definición de arquitecturas de redes neuronales profundas así como el entrenamiento de los modelos empleados. TensorFlow permitió además aprovechar la aceleración por GPU en el entrenamiento.

- **Scikit-fuzzy**: es una librería especializada en lógica difusa que proporciona herramientas para la construcción de sistemas de inferencia difusa y algoritmos de *clustering* difuso [38]. Fue utilizada en la implementación del enfoque alternativo basado en *fuzzy clustering* para la asignación de los APs.

Además de estas, se utilizaron librerías adicionales como, por ejemplo, **time** y **memory_profiler**, destinadas a la medición del tiempo de ejecución y del uso de memoria durante los experimentos, ya que representan aspectos clave para el análisis de eficiencia de los modelos implementados.

3.2.3 Otras herramientas

Además de los entornos de programación y las librerías específicas de Python, durante el desarrollo del proyecto se emplearon diversas herramientas complementarias que facilitaron la gestión, documentación y presentación de resultados. A continuación, se describen brevemente:

- **Git y GitHub**: Git es un sistema de control de versiones distribuido que permite registrar y gestionar cambios en archivos a lo largo del tiempo, facilitando el trabajo colaborativo y la trazabilidad de modificaciones [39]. GitHub complementa a Git ofreciendo una plataforma en la nube para el almacenamiento de repositorios, control de versiones, integración continua y colaboración en proyectos distribuidos. En este proyecto, Git se empleó para gestionar de manera estructurada los cambios en el código fuente, mientras que GitHub sirvió como repositorio remoto para la conservación y sincronización de todos los archivos correspondientes al desarrollo.

- **LaTeX (Overleaf)**: LaTeX es un sistema de preparación y maquetación de documentos basado en macros de TeX, orientado a la producción de textos técnicos y científicos de alta calidad tipográfica [40]. Permite gestionar de forma eficiente referencias bibliográficas, ecuaciones, índices y estructuras jerárquicas de documentos. Se utilizó a través de la plataforma online de Overleaf, que facilitó la edición colaborativa y la integración de bibliografía en formato BibTeX, así como la generación automática de figuras, tablas y ecuaciones complejas.

- **Jira**: Jira es una plataforma de gestión de proyectos y seguimiento de incidencias desarrollada inicialmente por Atlassian para metodologías ágiles [41]. Ofrece funcionalidades como la creación de tableros Kanban, planificación basada en historias de UE, y generación de informes de progreso. Durante el desarrollo del proyecto, se utilizó para planificar iteraciones, gestionar tareas pendientes y documentar el avance a través de tableros Kanban y diagramas de Gantt.

- **Power BI**: Power BI es un servicio de análisis empresarial proporcionado por Microsoft que permite conectar múltiples fuentes de datos, transformarlos y visualizar la información mediante paneles interactivos y reportes dinámicos [42]. Su objetivo es facilitar la toma de decisiones basada en datos. En este proyecto, Power BI se utilizó para la creación de informes interactivos para visualizar los resultados de las evaluaciones de los modelos considerados en diferentes escenarios, permitiendo así analizar de manera gráfica y accesible su rendimiento.

3.3 Organización del entorno de trabajo

Con el objetivo de mantener un desarrollo ordenado y facilitar la trazabilidad de los avances, el entorno de trabajo del proyecto se estructuró de manera sistemática tanto en el equipo local como en el repositorio remoto en GitHub. La organización general se basó en separar claramente los distintos componentes del proyecto: los códigos fuente, la documentación y los resultados generados.

Los *scripts* para la generación de simulaciones y de modelos de aprendizaje se agruparon en una carpeta de código. No obstante, se separaron los *scripts* de simulaciones y los de modelos en subcarpetas específicas para mejorar la claridad y el mantenimiento del proyecto. Los resultados obtenidos en las evaluaciones de eficiencia y rendimiento de los modelos se almacenaron, de nuevo, en subcarpetas diferenciadas dentro de una carpeta principal de resultados.

La gestión de las dependencias de Python se formalizó mediante la preparación de un archivo

requirements.txt, que recoge las librerías necesarias para la ejecución de los scripts. Además, la documentación del proyecto, incluyendo la memoria, el anteproyecto, la bibliografía y las imágenes utilizadas, se almacenó en la carpeta *Documents*.

La Figura 3.1 muestra un esquema de la organización general del entorno de trabajo:

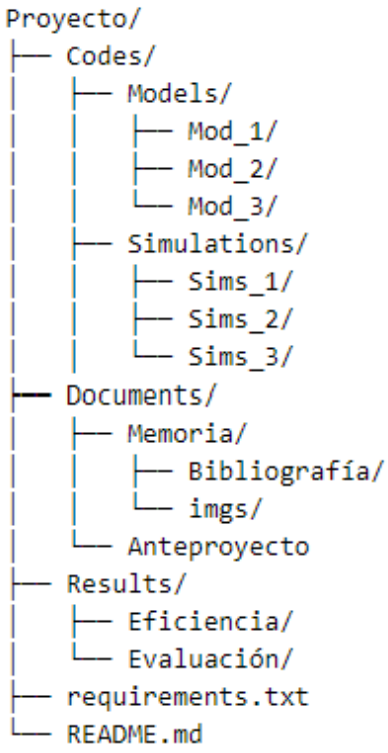


Figura 3.1: Estructura general del entorno de trabajo del proyecto. *(Generación propia)*

Cada sistema operativo utilizado (Windows y Ubuntu) contó con su propio entorno virtual independiente, adaptado a las particularidades de cada plataforma (versiones de Python, compatibilidad con TensorFlow GPU, etc.). Esta separación aseguró la coherencia en la ejecución de los experimentos y la portabilidad del proyecto entre entornos.

Esta organización, junto con el uso de control de versiones mediante Git, contribuyó a mantener un flujo de trabajo ágil y reproducible, facilitando la evolución ordenada del proyecto ya desde sus fases iniciales.

Capítulo 4

Desarrollo de la solución

En este capítulo se presenta el desarrollo de la solución propuesta para abordar el problema de asignación de punto de accesos (APs) en sistemas *Cell-Free Massive MIMO*. A partir de los conceptos teóricos y los modelos analizados en el Capítulo 2, se detalla la implementación del enfoque principal basado en Deep Contextual Bandits (DCB), así como de las aproximaciones adicionales, desarrolladas con fines comparativos.

Se describe el proceso de construcción de los modelos, diseño de la función de recompensa y tratamiento de los datos simulados. Además, se justifica el uso de las diferentes versiones de los modelos y se presentan las modificaciones progresivas que permitieron mejorar sus rendimientos.

El objetivo final del capítulo es mostrar el camino recorrido desde la teoría a la práctica, proporcionando una visión clara del enfoque adoptado.

4.1 Modelización del problema

En esta sección se presenta el planteamiento computacional del problema con el objetivo de recordar los aspectos expuestos anteriormente y establecer el contexto necesario para la implementación de las soluciones. Esta modelización sienta las bases para las secciones siguientes, en las que se detallan tanto la estructura del modelo DCB como de las implementaciones adicionales desarrolladas.

Como vimos previamente, en este proyecto se aborda el problema de asignación de APs a UEs en el contexto de sistemas *Cell-Free Massive MIMO*. A lo largo del Capítulo 2, se introdujeron los fundamentos teóricos y el estado del arte y, a continuación, se retoma el problema desde un enfoque computacional, planteándolo como una tarea de toma de decisiones basada en

contexto.

En este enfoque, cada UE representa una parte independiente del problema, donde el objetivo es determinar qué subconjunto de APs debería ser activado para maximizar su *rate* individual. Tal y como se describe en trabajos recientes como [29], aunque el objetivo es maximizar el *sum-rate* global, puede lograrse maximizando los *rates* individuales de cada uno de los UEs. Esto es posible porque el rendimiento de cada UE puede modelarse de forma independiente. Su tasa depende únicamente de su propia asignación de APs, y no de las asignaciones de otros UEs.

Para ello, el modelo comienza con un vector de contexto que contiene información como la ganancia del canal y la interferencia, I, causada por otros UEs en cada conexión. A partir de este, el modelo debe seleccionar una acción, codificada como un vector binario, indicando qué APs servirán al UE.

La calidad de cada acción se evalúa mediante una función de recompensa basada en el *rate* individual, R_k, alcanzada por el UE k, que se define, siguiendo la formulación de [29], como se indicó en la ecuación 2.1.

Este valor de R_k es el que sirve como base para la definición de la recompensa en los modelos de aprendizaje desarrollados.

4.2 Modelo basado en Deep Contextual Bandits

Para llevar a cabo esta tarea de asignación de APs, la idea es entrenar una red neuronal profunda. Su objetivo es calcular la recompensa que se obtendrá al conectarse a un subconjunto específico de APs basándose en el contexto que se ha observado y, de este modo, seleccionar el mejor subconjunto entre todos.

Esto se implementó mediante una arquitectura de tipo *actor-crítico*, en la que el *actor* genera las acciones a partir del contexto, mientras que el *crítico* las evalúa en función de la recompensa obtenida. Este diseño permite una actualización continua del modelo, lo que ayuda a crear un equilibrio entre exploración y explotación. Además, esta flexibilidad hace que los modelos de DCB sean una opción especialmente adecuada para entornos complejos y dinámicos como son los sistemas *Cell-Free Massive MIMO*.

Como paso previo a la construcción del contexto, se realiza un filtrado inicial de los APs dis-

ponibles. Para cada UE, se seleccionan únicamente las M_i APs más cercanas como candidatas a servirle, descartando el resto. Este filtrado permite simplificar la construcción del vector de entrada, centrándose en aquellos APs con mayores probabilidades de resultar relevantes, ya que los APs más alejadas presentan una menor ganancia de canal y una mayor interferencia esperada debido al efecto negativo de la distancia sobre ambos factores. Con este filtrado se simplifica el espacio de búsqueda de acciones, evitando la evaluación de combinaciones que, de otra forma, serían descartadas por el propio modelo.

La ganancia del canal entre el UE k y una AP i se calcula, de acuerdo a [25], como:

$$\beta_{i,k} = \mathrm{PL}_{i,k}^{-1}, \tag{4.1}$$

donde $\mathrm{PL}_{i,k}$ es la pérdida de trayectoria, que depende principalmente de la distancia entre ambos dispositivos. Este valor integra tanto la potencia de transmisión usada por el UE como las pérdidas asociadas a la propagación de las ondas electromagnéticas y a otros factores como el *shadowing*. El cálculo de la pérdida de trayectoria sigue el modelo descrito en [11].

Por otro lado, la interferencia depende de diversos factores relacionados con la transmisión simultánea de otros UEs. Como aproximación, en este trabajo se estima el grado de interferencia que percibe el UE k en el AP i en función de las ganancias de los demás UEs hacia dicho AP. Así, cuanto más próximos estén otros UEs, mayor será la interferencia percibida en comparación con la ganancia del UE que se está evaluando, y viceversa. En general, la potencia de las interferencias que recibe un UE k en el AP i, se estima como:

$$\mathrm{I}_i^{(k)} = \sum_{j \neq k} \beta_{i,j}, \tag{4.2}$$

donde el índice j hace referencia a UEs conectados al mismo AP i.

Una vez seleccionado el subconjunto de APs candidatos para cada UE, se construye un vector de contexto que representa el estado del entorno desde su perspectiva. Este vector se genera a partir de las simulaciones (para cada escenario) y está compuesto por tres elementos clave: el identificador del AP, la ganancia del canal del UE hacia el mismo, y la interferencia causada por otros UEs conectados a ese AP. El vector de contexto para el UE k se obtiene, por tanto, concatenando esta información para los M_k APs preseleccionados para ese UE.

A continuación, presentamos un ejemplo que ilustra este paso. Dado un determinado escenario, para el UE 0 simulado podríamos tener los siguientes datos:

UE	AP	Ganancia	Interferencia	Distancia
0	3	0.82	0.11	0.01
0	7	0.65	0.23	0.13
0	5	0.57	0.07	0.15

Tabla 4.1: Ejemplo de datos simulados para un UE con tres APs candidatos.

En este caso, tendríamos el vector de entrada correspondiente:

$$\mathbf{x} = [3,\ 0.82,\ 0.11,\ 7,\ 0.65,\ 0.23,\ 5,\ 0.57,\ 0.07]$$

De este modo, llegamos a la dimensión del vector de entrada para cada UE, que sería $3 \cdot M_k$. Para mantener la consistencia en la red al variar M_k, se aplica un *padding* con ceros hasta alcanzar una dimensión fija M_{max} que, en este trabajo, se ha establecido a $M_{\text{max}} = 20$. Así el modelo es capaz de procesar la información correspondiente a cualquier UE como un vector de dimensión $3 \cdot M_{\text{max}}$.

Siguiendo con el ejemplo anterior, obtendríamos el siguiente vector de entrada:

$$\mathbf{x}_{\text{final}} = [3,\ 0.82,\ 0.11,\ 7,\ 0.65,\ 0.23,\ 5,\ 0.57,\ 0.07,\ \underbrace{0, \ldots, 0}_{3(M_{\text{max}}-3)\text{ elementos}}\ ]$$

Tras la predicción, se aplica un umbral para binarizar la salida, que es un vector donde cada elemento representa el grado de activación asociado a un determinado AP, y obtener un vector de acciones. En él, los valores de 1 indican los APs a conectarse, y los de 0 corresponden a APs descartados. Los elementos del vector de salida correspondientes a posiciones de relleno son enmascarados y fijados a cero para evitar decisiones sobre estos APs no reales.

Durante el entrenamiento, los vectores se agrupan en lotes (*batches*) de tamaño fijo B, lo que da lugar a entradas de dimensión $(B, 3 \cdot M_{\text{max}})$ y salidas de dimensión (B, M_{max}). Esta estructura fija de las entradas y salidas permite aprovechar operaciones matriciales estándar durante el entrenamiento, sin necesidad de diseñar arquitecturas específicas para gestionar diferentes tamaños de entrada en función del número de APs candidatos de cada UE. De este modo, el modelo resulta fácilmente escalable a distintos tamaños de escenarios simplemente ajustando el valor de M_{max}.

Cabe destacar que, para garantizar una mayor estabilidad durante el entrenamiento y facilitar

la comparación entre diferentes escenarios, tanto los valores de ganancia como los de interferencia son normalizados al generar el conjunto de datos. En ambos casos, se emplea una normalización simple dividiendo por el valor máximo del escenario. Así conseguimos que las entradas se encuentren en el intervalo $[0, 1]$. Estas normalizaciones se pueden expresar matemáticamente de la siguiente forma:

$$\tilde{\beta}_{i,k} = \frac{\beta_{i,k}}{\max\limits_{\substack{i\in\{1,\dots,L\}\\ k\in\{1,\dots,K\}}} \beta_{i,k}}, \tag{4.3}$$

$$\tilde{\mathrm{I}}_i^{(k)} = \frac{\mathrm{I}_i^{(k)}}{\max\limits_{\substack{i\in\{1,\dots,L\}\\ k\in\{1,\dots,K\}}} \mathrm{I}_i^{(k)}}, \tag{4.4}$$

Esta transformación ayuda a mejorar la eficiencia del entrenamiento y la generalización del modelo.

En la Figura 4.1, se muestra un esquema del flujo de información en el modelo para un UE, desde los datos simulados hasta la decisión final generada por la red neuronal.

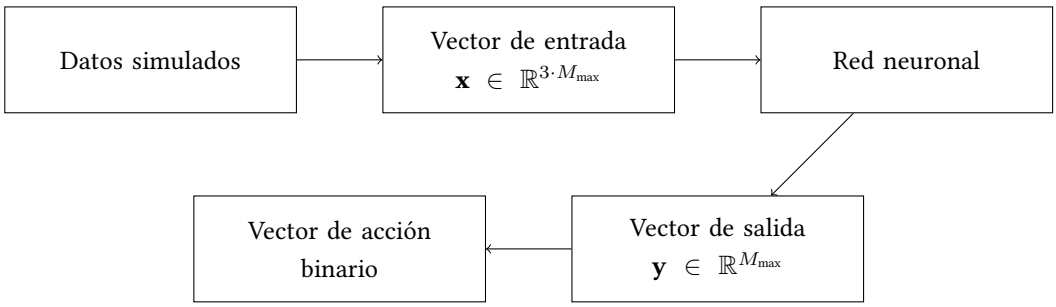


Figura 4.1: Flujo de datos desde las simulaciones hasta la acción generada por el modelo.

4.2.1 Diseño de la función de recompensa

Una vez explicado el formato de entrada y salida, así como el flujo de datos, pasamos a su procesamiento en el modelo a través de la función de recompensa. Esta define el criterio con el que el agente evalúa la calidad de sus decisiones, por lo que su diseño es un aspecto clave.

En este proyecto, el componente central sobre el que se construyen las estrategias de asignación es el *sum-rate* del escenario, R, el cual, como se vio previamente, se define como la suma de las tasas individuales obtenidas por cada UE tras activar su correspondiente conjunto de APs. Para un determinado UE k, su tasa individual R_k se calcula, siguiendo el modelo de

capacidad de canal de Shannon [43], como se detalló previamente en la ecuación (2.1) del Capítulo 2. Esta tasa recoge tanto las ganancias de canal como las interferencias experimentadas en función de las asignaciones realizadas por el modelo.

En el caso del modelo DCB, la función de recompensa se basa directamente en esta tasa de transmisión individual R_k. De este modo, el objetivo del modelo es seleccionar las combinaciones de APs activadas para cada UE de forma que su correspondiente R_k quede maximizado. Esta formulación permite abordar el problema de forma independiente por UE, simplificando el proceso de aprendizaje y adaptación a los contextos dinámicos generados en cada escenario simulado.

Pero además del *sum-rate*, se introdujeron factores de regularización que modifican el valor de la recompensa en base al comportamiento reciente del agente del siguiente modo:

1. **Factor de escalado**: penaliza configuraciones con un número muy bajo de APs activas, premiando aquellas que utilizan un número razonable de enlaces. Se calcula como:

$$scaling\_factor = \frac{\sum_i a_k(i)}{M_k} \tag{4.5}$$

2. **Penalización por acción repetida**: si el agente selecciona la misma acción en pasos seguidos, se le aplica una penalización para evitar estancarse en una única asignación y favorecer la diversidad de decisiones a lo largo del tiempo.

3. **Penalización por asignación insuficiente**: si el número de APs activas es inferior a la mitad del total disponible se aplica una penalización. Esta medida busca evitar asignaciones excesivamente conservadoras, en las que el modelo active muy pocos APs, reduciendo innecesariamente la capacidad de transmisión. El umbral de la mitad se selecciona como un compromiso razonable para asegurar un nivel mínimo de recursos asignados sin saturar el sistema.

$$\text{penalty} = \begin{cases} -5 & \text{si } \sum_i a_k(i) < 0.5 \cdot M_k \\ 0 & \text{en otro caso} \end{cases} \tag{4.6}$$

Considerando todos los factores descritos, la función de recompensa final empleada en los modelos basados en pérdida se corresponde con la definida previamente en la ecuación (2.2) del Capítulo 2, donde se combina el *sum-rate* total con los factores de regularización aplicados.

Cabe destacar que esta formulación, al no requerir conocimiento global del entorno, es computacionalmente eficiente y aplicable directamente en escenarios simulados como los empleados en este trabajo.

4.2.2 Arquitectura

Sabemos que en el modelo usado se distinguen dos componentes principales: el actor, que genera la acción en base al contexto, y el crítico, encargado de evaluar la acción en base a su recompensa esperada. El código 4.1 muestra la arquitectura utilizada para ambas redes:

```
state_dim = 3
action_dim = 20      # M_max

# Arquitectura del actor
def build_actor(state_dim, action_dim):

    model = keras.Sequential([
        layers.Input(shape=(state_dim * action_dim,)),
        layers.Dense(128, activation='relu',
    kernel_regularizers=regularizers.l2(1e-4)),
        layers.Dropout(0.2),
        layers.Dense(64, activation='relu',
    kernel_regularizers=regularizers.l2(1e-4)),
        layers.Dropout(0.2),
        layers.Dense(32, activation='relu',
    kernel_regularizers=regularizers.l2(1e-4)),
        layers.Dropout(0.2),
        layers.Dense(action_dim, activation='sigmoid')
    ])
    return model

# Arquitectura del crítico
def build_critic(state_dim, action_dim):

    model = keras.Sequential([
        layers.Input(shape=(state_dim * action_dim + action_dim,)),
        layers.Dense(128, activation='relu',
    kernel_regularizers=regularizers.l2(1e-4)),
        layers.Dropout(0.2),
        layers.Dense(64, activation='relu',
    kernel_regularizers=regularizers.l2(1e-4)),
        layers.Dropout(0.2),
        layers.Dense(32, activation='relu',
    kernel_regularizers=regularizers.l2(1e-4)),
```

```
29          layers.Dropout(0.2),
30          layers.Dense(1)
31      ])
32      return model
```

Listing 4.1: Arquitectura del modelo actor-crítico

Se aprecia en las líneas 8-15 y 23-30 que ambas redes comparten una estructura base compuesta por tres capas densas, con tamaños 128, 64 y 32 respectivamente. Todas ellas utilizan funciones de activación ReLU y regularización L_2[1], a lo que se añaden capas *Dropout* con ratio 0,2 tras cada capa para reducir el sobreajuste y mejorar la generalización.

La entrada del actor tiene dimensión $3{\cdot}M_{\text{max}}$, correspondiente al vector de contexto del UE. Mientras que el crítico recibe una entrada de dimensión $4{\cdot}M_{\text{max}}$, ya que añade a lo anterior el vector binario de acción asociado.

Después, encontramos en la línea 16 la salida del actor, que toma la representación del contexto generada y produce un vector de salida de dimensión M_{max}, donde cada elemento representa el grado de activación asociado a un determinado AP. Esta capa utiliza una activación de tipo sigmoide para asegurar que cada valor esté en el intervalo [0,1].

Finalmente, en la línea 31, tenemos la capa de salida del crítico. A partir de la representación del contexto y la acción, estima la recompensa esperada para la acción seleccionada por el actor. Su salida es un escalar correspondiente al valor de la función objetivo.

Cabe destacar que, aunque ambas redes comparten la arquitectura base, se entrenan con objetivos distintos y no comparten pesos entre sí. Esta separación permite especializar cada red en su tarea correspondiente dentro del marco actor-crítico.
En cuanto al diseño de la arquitectura, se exploraron varias configuraciones mediante pruebas empíricas, evaluando distintas profundidades, funciones de activación y ratios de *dropout*. Finalmente, se seleccionó la estructura actual por su buen equilibrio entre rendimiento y estabilidad. Este proceso se llevó a cabo utilizando las herramientas de ajuste ofrecidas por la biblioteca Keras, sin realizar una búsqueda sistemática exhaustiva, pero validando experimentalmente que la arquitectura elegida ofrecía buenos resultados de convergencia y generalización.

[1] La regularización L_2 añade un término $\lambda \Sigma_i w_i^2$ al valor de la pérdida total, donde w_i son los pesos de la red y λ es el coeficiente de regularización [44].

Una vez definida la arquitectura de las dos redes, ambos modelos son compilados[2] con el optimizador `Adam`, muy utilizado en tareas de aprendizaje profundo por su capacidad para adaptarse dinámicamente al ritmo de aprendizaje de cada parámetro [45]. En concreto, el actor se entrena con un *learning rate* de 0,01, valor que busca favorecer la exploración del espacio de políticas al inicio del entrenamiento. Mientras, el crítico utiliza un *learning rate* de 0,001, con el fin de garantizar una convergencia más estable.

4.2.3 Entrenamiento

El entrenamiento del modelo DCB se basa en la interacción del agente con diversos escenarios, siguiendo un enfoque típico del aprendizaje por refuerzo. Los principales parámetros usados durante el entrenamiento son:

- **Número de episodios:** 200. Cada episodio se corresponde a una interacción con un nuevo escenario simulado. En cada uno de estos escenarios, se genera un número aleatorio de APs candidatos por UE M_k, distribuciones aleatorias de UEs y APs totales en el escenario, se consideran diferentes valores de K y L, permitiendo así una evaluación en entornos variables.
- **Número de *epochs* por episodio:** 10. Cada escenario se entrena durante 10 ciclos completos para que el modelo se pueda ajustar a las condiciones del entorno antes de pasar al siguiente episodio.
- **Capacidad del *Replay Buffer*:** 5000. Se almacena una memoria de interacciones pasadas para reutilizar ejemplos y estabilizar el entrenamiento.
- **Tamaño del *batch*:** 32. Cada actualización de pesos se realiza con un conjunto de 32 ejemplos tomados de la memoria de experiencias (*Replay Buffer*).

El entrenamiento se hizo a través de una estrategia personalizada basada en una función *train step* manual en lugar del método clásico `model.fit()`, ya que la primera permitía un mayor control sobre el proceso de optimización y facilitaba la incorporación de elementos propios de los algoritmos actor-crítico, como la retroalimentación diferenciada para cada red y la actualización del *buffer* de experiencias.

El entrenamiento se realiza a lo largo de múltiples episodios, cada uno con una simulación aleatoria del escenario. Para cada UE, el actor calcula el vector de acción, que es binarizado y,

[2] En el contexto de aprendizaje automático con *frameworks* como Keras o TensorFlow, compilar un modelo significa configurarlo con el optimizador, la función de pérdida y las métricas antes de comenzar el entrenamiento.

posteriormente, evaluado a través de la función de recompensa descrita en la sección 4.2.1.

Esta información (estado, acción y recompensa) se almacenan en el *buffer* de experiencias desde donde se extraen lotes (*bachtes*) para actualizar los modelos. El crítico se entrena para aproximar la recompensa observada, mientras que el actor se actualiza para maximizar la estimación del crítico. Además, este último cuenta también con un término de entropía para favorecer la exploración. Es decir, incentiva al actor a seguir probando distintas acciones durante el entrenamiento en lugar de centrarse demasiado pronto en una única estrategia.

El código 4.2 muestra un resumen del proceso de entrenamiento:

```
for episode in range(num_episodes):
    # Se genera el escenario
    generate_scenario()

    # Almacenamiento de la experiencia en el buffer
    for user in users:
        action = actor.predict(state)
        binary_action = binarize(action)
        reward = calculate_reward(state, binary_action)
        replay_buffer.add(state, binary_action, reward)

    # Selección del conjunto (batch) para el episodio de
    entrenamiento (si hay suficientes experiencias)
    if replay_buffer.size() >= batch_size:
        states, actions, rewards = replay_buffer.sample(batch_size)

        # Actualización del crítico
        with tf.GradientTape() as tape:
            q_values = critic(concat(states, actions))
            critic_loss = mse(rewards, q_values)
        apply_gradients(critic, critic_loss)

        # Actualización del actor
        with tf.GradientTape() as tape:
            predicted_actions = actor(states)
            actor_loss = -critic(concat(states, predicted_actions))
            actor_loss += entropy(predicted_actions)
        apply_gradients(actor, actor_loss)
```

Listing 4.2: Resumen del bucle de entrenamiento

4.2.4 Evaluación

Aunque la generación de escenarios se explicará con mayor detalle en el siguiente capítulo, es importante introducir brevemente el enfoque seguido para generar los datos usados durante el entrenamiento y la validación del modelo.

Primero, es necesario mencionar que los datos generados para cada escenario ya se encuentran en su formato final antes de ser procesados por la red, es decir, como vectores que agrupan información sobre los identificadores, las ganancias de canal y las interferencias hacia los APs candidatos para cada UE. Estos datos se almacenan y gestionan únicamente durante el entrenamiento, ya que son generados dinámicamente en cada episodio.

En cuanto al entrenamiento, en cada episodio se genera un nuevo escenario simulado con una distribución aleatoria de UEs y APs, dentro de los rangos establecidos para los experimentos. Esto permite que el modelo observe una gran variedad de configuraciones, favoreciendo la generalización y evitando que memorice un entorno específico. En total, se simularon 500 episodios de entrenamiento, lo que proporciona una cantidad suficientemente amplia de ejemplos para el ajuste de los modelos.

Además, se generaron 20 escenarios de validación antes de comenzar el entrenamiento, que se mantuvieron fijos a lo largo de todo el proceso para asegurar una evaluación justa. En cada uno de estos escenarios se simulan entre 5 y 45 UEs. El rendimiento del modelo se calcula como la media del *sum-rate* alcanzado entre todos los UEs y escenarios. Esto garantiza una evaluación coherente y evita que el modelo obtenga mejores resultados simplemente por haber visto previamente los datos. Además, la validación se realiza periódicamente durante el entrenamiento, permitiendo monitorizar la evolución y detectar sobreajuste.

Para esta evaluación se usaron diferentes técnicas como el uso de *Dropout* y regularización $L2$, que ya se comentaron en secciones previas. Asimismo, se registraron de manera iterativa tanto las recompensas por episodio como la pérdida de las redes. Estos valores se almacenaron en archivos .csv y se graficaron para visualizar la convergencia del modelo. Se utilizó además, en el caso de la recompensa, una media móvil para suavizar las fluctuaciones y facilitar la visualización de las tendencias.

Como se muestra en la Figura 4.2, hay una evolución ascendente de la recompensa, lo que indica una mejora en la capacidad del modelo para seleccionar buenas acciones. La evaluación sobre el conjunto de validación se hizo con recompensas normalizadas para poder comparar entre distintos escenarios y modelos.

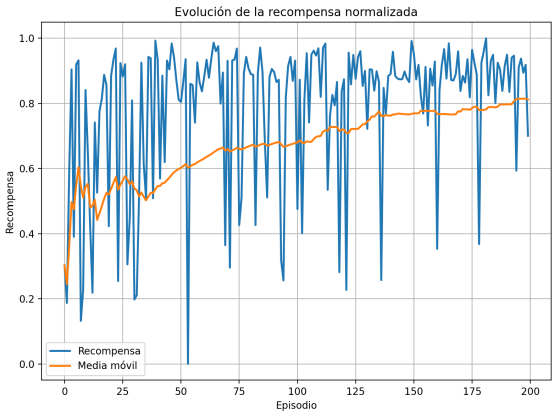
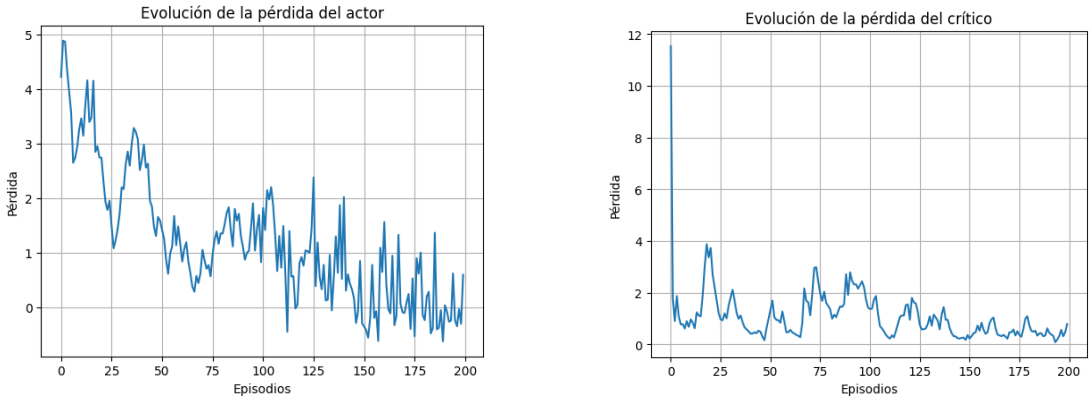


Figura 4.2: Evolución de las recompensas durante el entrenamiento. *(Generación propia)*

Finalmente, para evitar entrenamientos excesivos, se empleó una estrategia de *early stopping* que se activaba automáticamente cuando la recompensa media de los últimos episodios se estabilizaba por debajo de un umbral de variación. Esta técnica no solo sirve para evitar el sobreajuste, sino que también reduce el coste computacional del proceso de entrenamiento.



(a) (b)

Figura 4.3: Evolución de las funciones de pérdida durante el entrenamiento de las redes actor (a) y crítico (b). *(Generación propia)*

Podemos observar en la Figura 4.3 la evolución de las funciones de pérdida del actor y del crítico durante el entrenamiento del modelo. En la subfigura (a), para el actor, se observa una pérdida con alta variabilidad, sobre todo en las primeras etapas. Esto puede estar causado por el carácter exploratorio del actor. Aunque la tendencia general es decreciente, el actor experimenta picos y valles que reflejan la adaptación de la política de asignación.

Sin embargo, en la subfigura (b), para el crítico, se aprecia un pico inicial seguido de una caí-

da clara y una posterior estabilización. Esto se debe a que el crítico aprende a aproximar el valor esperado de las acciones en función de la política actual, y eso hace que converja más rápidamente al tener una referencia de entrenamiento más directa.

En conjunto, las gráficas reflejan un comportamiento satisfactorio del modelo actor-crítico. Mientras que el crítico se estabiliza con mayor rapidez, el actor sigue ajustándose en función de las evaluaciones del crítico.

4.3 Aproximaciones adicionales

Aunque el enfoque princial del proyecto es el modelo basado en DCB, también se exploraron otras opciones con el fin de comparar su rendimiento y evaluar posibles soluciones alternativas. Estas aproximaciones permiten abordar el problema desde diferentes perspectivas y sirven de referencia para validar la eficacia del enfoque principal. En esta sección, se describen los modelos alternativos implementadas en el proyecto.

4.3.1 Modelos basados en pérdida

Estos modelos son una alternativa simple y efectiva para abordar el problema de asignación de APs. A diferencia de los modelos basados en RL, no necesitan un entorno de entrenamiento para gestionar las interacciones de un agente ni políticas de exploración, sino que se entrenan directamente para minimizar una función de pérdida personalizada, en este caso basada en funciones derivadas de la expresión del *rate* individual, R_k.

En este caso, se implementó una red neuronal densa, similar a las del modelo DCB, que recibe como entrada un tensor de dimensión $(20, 2)$, correspondiente a las ganancias de canal y las interferencias para cada uno de los $M_{max} = 20$ APs candidatos. La salida, igual que en el modelo DCB, es un vector de probabilidad de activación de cada AP sobre el que se aplica un umbral para obtener un resultado binario.

Para la pérdida se diseñaron dos funciones inspiradas en trabajos sobre optimización del *throughput* (velocidad efectiva de transmisión) y formulaciones basadas en recompensa acumulada [15, 46], con el fin de comprobar qué versión obtenía mejores resultados. Así como el efecto que conlleva modificar la función de pérdida aunque sean aparentemente equivalentes. La primera variante está basada en la maximización directa del *rate* (función objetivo negativa: $-R_k$) y, la otra, en la minimización de su inverso ($1/R_k$). En ambos casos, se utilizó la misma arquitectura de red.

Las formulaciones de las funciones de pérdida en cuestión son las siguientes:

$$Loss_{neg} = -\sum_{k} R_k, \tag{4.7}$$

$$Loss_{inv} = \sum_{k} \frac{1}{R_k}, \tag{4.8}$$

donde R_k se define en la ecuación 2.1.

Además, para incentivar la activación de un número razonable de APs, igual que se realizó en el modelo de DCB, se incorporaron al objetivo dos componentes adicionales:

- Un término de escalado que favorece la activación de una fracción suficiente de APs.
- Una penalización por activar menos del 50% de los APs candidatos.

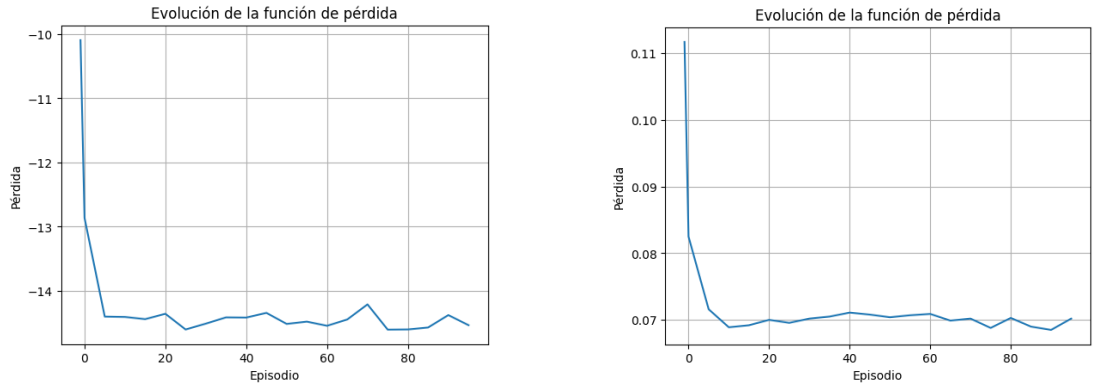
En cuanto al entrenamiento, se muestra a continuación el bloque central del código donde se definen las posibles funciones de pérdida derivadas del *sum-rate* de los UEs y calculadas de acuerdo a las ecuaciones 4.7 y 4.8:

```
# Parámetros
# numActiveAps = M_i
# N0 = 1e-6
penalty_weight = 0.5

@tf.function
def train_step(model, x, inv):
    with tf.GradientTape() as tape:

    # Predicción de las acciones
    y_pred = model(x, training=True)

    gain = tf.cast(x[:, :, 0], tf.float32)
    interference = tf.cast(x[:, :, 1], tf.float32)
    selected_gain = tf.reduce_sum(y_pred * gain, axis=1)
    selected_interference = tf.reduce_sum(y_pred * interference,
    axis=1)

    # Cálculo del sum-rate
    sum_rates = tf.math.log(1 + (selected_gain /
    (selected_interference + N0))) / tf.math.log(2.0)

        # Construcción de la función de pérdida según la variante
    escogida
```

```
        if inv:
            loss = tf.reduce_sum(1 / (sum_rates + N0))
        else:
            loss = tf.reduce_sum(-sum_rates)

        penalty = tf.nn.relu(tf.cast(numActiveAPs, tf.float32) *
    penalty_weight - tf.reduce_sum(y_pred))
        loss += penalty_weight * penalty

# Cálculo y aplicación de los gradientes
gradients = tape.gradient(loss, model.trainable_variables)
model.optimizer.apply_gradients(zip(gradients,
    model.trainable_variables))
return loss
```

Listing 4.3: Entrenamiento basado en minimizar función de pérdida

Los resultados del entrenamiento para ambas variantes los podemos observar en la Figura 4.4. En ella, podemos ver una gran disminución inicial de la función de pérdida y una posterior estabilización en ambos casos. Esto indica que los modelos consiguen aprender de forma eficiente y convergen adecuadamente. Si bien en el modelo que usa la función de pérdida basada en el negativo del *sum-rate* se aprecia mayor variabilidad (Figura 4.4(a)), esto podría deberse a una mayor sensibilidad del modelo a los datos de entrada o a la propia naturaleza de la función de pérdida utilizada. Aun así, el comportamiento general sigue siendo bastante estable y satisfactorio en ambos casos.



(a) (b)

Figura 4.4: Evolución de la función de pérdida durante el entrenamiento de los modelos *loss-based* negativo (a) e inverso (b). *(Generación propia)*

Para finalizar, se recoge un pequeño resumen de las principales fortalezas y limitaciones de los modelos basados en pérdida como enfoque alternativo.

Fortalezas:

- Permiten un entrenamiento directo y guiado hacia el *sum-rate*.
- Presentan una arquitectura simple y rápida de entrenar.
- No requieren interacción con un entorno dinámico.

Limitaciones:

- No exploran nuevas políticas ni ajustan su comportamiento en función de la experiencia.
- Su rendimiento puede empeorar ante cambios no vistos o anticiapados en los datos.
- Presentan cierta falta de adaptabilidad dinámica, lo que los limita en entornos cambiantes.

4.3.2 *Clustering* difuso

Como segunda alternativa, se exploró una aproximación no supervisada y de bajo coste computacional. Esta consiste en agrupar APs en torno a cada UE, que representarían los centros de los clústeres, sin necesitar un proceso de entrenamiento supervisado. De este modo, probamos un método directo de agrupación (asignación) que evita el uso de redes neuronales. Esta estrategia resulta especialmente atractiva dado que permite incorporar criterios físicos como la ganancia de los canales o el *sum-rate* sin recurrir a funciones de pérdida ni retropropagación.

El funcionamiento del algoritmo de *clustering* difuso se explicó en detalle en el Capítulo 2. Además, conviene recordar que, en el contexto de la asignación de APs, se construye una matriz que refleja la relación entre UEs y APs a partir de los valores agrupados sacados del escenario simulado. Esta matriz contiene las ganancias de canal de los UEs y las interferencias correspondientes para cada AP en el sistema. Todos estos datos nos proporcionan información relevante sobre la distancia de los UEs a los APs y permiten el cálculo de los *sum-rate* correspondientes para sub-grupos de APs activados.

Teniendo en cuenta esta información, se puede aplicar el algoritmo Fuzzy C-Means, que nos devuelve los centros de los clústeres (UEs) y la matriz de pertenencias u_{ik}, que indica el grado de pertenencia de la AP_i al clúster del UE_k. A continuación, se calcula el *rate* potencial de conexión entre el UE y cada AP, y se combina con el grado de pertenencia para obtener una

métrica compuesta que guíe la selección de las mejores APs. El *rate* potencial se calcula a partir de la capacidad de canal de Shannon [43], como:

$$R_i^{(k)} = \log_2 \left(1 + \frac{\beta_{i,k} \cdot G_{\max}}{I_i^{(k)} \cdot G_{\max} + N_0} \right), \tag{4.9}$$

donde:

- $\beta_{i,k}$: ganancia de canal entre el AP_i y el UE_k,
- $I_i^{(k)}$: interferencia que afecta al UE_k desde otros UEs conectados al AP_i,
- G_{max}: ganancia máxima observada en el escenario.

Finalmente, se combinan los valores de pertenencia, u_{ik}, con las tasas potenciales, $R_i^{(k)}$, estimadas para cada AP_i y UE_k, obteniendo una métrica compuesta que refleja tanto la calidad de la conexión como la cercanía relativa entre el UE y el AP. Esta métrica se calcula como el producto de u_{ik} y $R_i^{(k)}$.

A continuación, se ordenan los APs candidatos según esta métrica compuesta y se seleccionan aquellos cuyo $R_i^{(k)}$ supera un umbral (*threshold*) predefinido. Este actúa como filtro para evitar asignaciones poco beneficiosas. La elección de su valor es especialmente crítica, ya que el rendimiento del sistema es muy sensible a él. Por ello, en lugar de fijarlo manualmente, se recurre a una búsqueda binaria (*bisección*) que permite encontrar un valor de *threshold* que maximice el rendimiento en cada escenario simulado de forma eficiente. Además, si ningún valor supera el umbral, se selecciona el AP con mayor $R_i^{(k)}$ como mínimo asegurado.

Cabe destacar que se multiplica por $G_{\max}$ en el cálculo de $R_i^{(k)}$ para escalar tanto la ganancia como la interferencia a una magnitud comparable y evitar que valores muy pequeños, provenientes de normalizaciones previas, afecten negativamente a la expresión del SINR. Esto facilita una mejor discriminación entre opciones durante el proceso de selección.

En el código 4.4, se muestra un resumen de este procedimiento recién explicado:

```
# n_clusters = Número de clusters
# m = Parámetro de difusividad

def fuzzy_clustering_aps(data, max_gain, n_clusters, n_aps,
    act_aps, threshold, noise, m=2):
    # Agrupación de los datos en 'clustering_data' y 'data'
    ...

```

```
    # Aplicación del algoritmo Fuzzy C-Means
    cntr, u, _, _, _, _, _ = fuzz.cluster.cmeans(
        clustering_data,
        c=n_clusters,
        m=m,
        error=0.005,
        maxiter=1000
    )

    # Agrupación de los datos originales (data) por UE
    grouped = data.groupby('UE_id')

    for ue_id, ue_data in grouped:

        ap_indices = ue_data.index.to_numpy() - (20 * ue_id)

        # Selección del clúster asignado al usuario y extracción del grado de pertenencia para cada AP
        ue_cluster = ue_id % n_clusters
        memberships = u[ue_cluster, ap_indices]

        gains = ue_data['user_gain'].to_numpy()
        interference = ue_data['interference_sum'].to_numpy()

        # Cálculo del SINR y del rate potencial para cada AP
        sinr = (gains * max_gain) / (interference * max_gain + noise)
        potential_rates = np.log2(1 + sinr)

        # Ordenación de APs candidatas según la métrica compuesta
        combined_scores = memberships * potential_rates
        sorted_aps = np.argsort(combined_scores)[::-1]

        # Selección de las mejores APs por encima del umbral
        selected_aps = []
        for ap_idx in sorted_aps[:act_aps]:
            if potential_rates[ap_idx] > threshold:
                selected_aps.append(ap_idx)
            if len(selected_aps) >= act_aps:
                break

        # En caso de no cumplir umbral, se selecciona la AP con mayor rate
        if not selected_aps:
            selected_aps = [np.argmax(potential_rates)]

```

```
        # Creación del vector final
        final_aps = np.zeros_like(memberships)
        final_aps[selected_aps] = 1
```

Listing 4.4: Implementación del algoritmo de *Fuzzy Clustering* para asignación de APs

De nuevo, finalizamos con un pequeño resumen de las principales fortalezas y limitaciones de este enfoque.

Fortalezas:

- Es fácil de implementar, a la vez que no requiere entrenamiento ni gestión compleja de hiperparámetros.
- Tiene bajo coste computacional.
- Permite controlar, en cierta medida, la cantidad de APs asignados a través del umbral.
- Puede adaptarse a varios criterios físicos.[3]

Limitaciones:

- No aprende progresivamente ni optimiza una función objetivo. Esto limita su flexibilidad y capacidad de adaptación en entornos muy dinámicos y cambiantes.
- Su rendimiento depende de la selección del umbral (a la que es extremadamente sensible) y del criterio de agrupamiento.

4.4 Resumen del desarrollo

Ahora que se completó la descripción del desarrollo de las distintas soluciones propuestas, se presenta una visión global de los enfoques explorados y su aplicabilidad al problema de asignación de APs en entornos *Cell-Free Massive MIMO*.

El modelo principal basado en DCB se implementó con una arquitectura sencilla y mecanismos de exploración-explotación que permiten aprender políticas de asignación eficientes. Se desarrollaron también dos aproximaciones adicionales: un modelo supervisado basado en funciones de pérdida personalizadas, y un algoritmo de *fuzzy clustering* adaptado a este tipo de escenarios. La primera estrategia permitió optimizar directamente la métrica deseada, y la

[3] Aunque inicialmente se valoró utilizar métricas como la distancia, finalmente el agrupamiento se realizó directamente con las ganancias de canal, lo que permite integrar de forma implícita tanto el efecto de la distancia como otras características del entorno.

segunda, simplificar el proceso mediante agrupación difusa.

En cuanto a los resultados de los entrenamientos, tanto el modelo de DCB como las dos variantes del basado en pérdida presentaron convergencias estables y, en el caso de la estrategia principal, una evolución creciente de la función de recompensa. Esto refuerza su validez como soluciones prácticas y escalables.

Para finalizar el capítulo con una comparación visual entre los modelos, en la Tabla 4.2 se presenta un resumen de las características más relevantes de cada uno de ellos.

Modelo	**Tipo**	**Entrenamiento**	**Exploración**	**Ventajas destacadas**
DCB (Actor-Crítico)	Deep Contextual Bandit	No supervisado, basado en retroalimentación	Sí	Adaptabilidad
Modelos *Loss-Based*	Supervisado	Minimiza función de pérdida (sum-rate o inversa)	No	Entrenamiento directo sobre la métrica objetivo
Fuzzy Clustering	No supervisado	No requiere entrenamiento	No	Simplicidad y control sobre el *threshold*

Tabla 4.2: Comparación de los enfoques propuestos.

4.5 Observaciones durante el desarrollo

Durante la implementación de los modelos, surgieron varios desafíos que requirieron ajustes específicos para garantizar la estabilidad del entrenamiento y la calidad de las asignaciones.

Uno de los primeros problemas detectados fue la disparidad de escalas entre la ganancia de canal y la interferencia, que provocaba un comportamiento errático del modelo durante las primeras pruebas. Para solucionarlo, se probaron distintas estrategias de normalización hasta que, finalmente, se aplicó una *min-max* entre [0, 1] a ambas variables antes de usarlas como entrada de los modelos, que estabilizó el entrenamiento y mejoró significativamente la convergencia de las redes.

Otro ajuste clave fue la elección del umbral de activación aplicado sobre la salida de los modelos. Inicialmente se utilizaron valores genéricos como 0.5, pero, al ver que el resultado era sensible a ese parámetro, se optó por distintos valores según el modelo: por ejemplo, en el caso del DCB, un umbral más bajo permite activar más APs por UE y compensar a su estrategia

más conservadora.

En el caso de la aproximación por *clustering* difuso, se observó que el rendimiento era extremadamente dependiente del umbral. Se probó inicialmente con valores fijos y, a continuación, con un umbral dinámico en función del número de UEs y/o APs en el escenario. Pero su sensibilidad llevó a implementar una búsqueda binaria por bisección, que ajusta el umbral en cada escenario para maximizar el *sum-rate*. Esta decisión tuvo un impacto notable en el rendimiento del modelo, que pasó de estar por debajo del aleatorio a competir con los modelos entrenados.

Finalmente, durante el entrenamiento del modelo DCB, también hubo problemas con la actualización de pesos mediante `model.fit`, que provocaba errores en los gradientes. Para solucionarlo, se reescribió el proceso utilizando una función personalizada `train_step()` con `GradientTape()` de TensorFlow. Esto permitió un control más detallado sobre el cálculo de la pérdida y la aplicación de los gradientes, lo que fue clave para que el modelo convergiera correctamente.

Capítulo 5

Análisis de resultados

En este capítulo se presentan los resultados obtenidos tras la evaluación de los distintos modelos. Se analizan tanto el modelo principal basado en DCB como las aproximaciones alternativas desarrolladas con fines comparativos.

Inicialmente se describen los escenarios utilizados en las simulaciones, seguido de una explicación del proceso de generación y obtención de los datos a partir de los mismos. A continuación, se detallan las métricas empleadas para evaluar el rendimiento de los modelos y se analiza su comportamiento con diferentes configuraciones.

Finalmente, se incluye una visualización de los resultados mediante un cuadro de mando interactivo desarrollado en Power BI.

5.1 Descripción de los escenarios de evaluación

Para evaluar el rendimiento de los distintos enfoques propuestos, se diseñaron múltiples escenarios simulados que reproducen entornos realistas de redes *Cell-Free Massive MIMO*, siguiendo como ejemplo los utilizados en [2]. De este modo, se puede observar cómo se comportan los modelos en diferentes configuraciones de red, variando parámetros como el número de punto de accesos (APs), el de dispositivo usuarios (UEs) o el tamaño del área del escenario.

Cada uno de estos escenario se genera con una función que sitúa los UEs y APs dentro de un área de cobertura que puede variar entre los $500m^2$ y $1500m^2$. La colocación de los APs puede ser aleatoria o seguir una distribución ordenada (en rejilla). Esto permite evaluar también el impacto de la geometría en la asignación. Por otro lado, las posiciones de los UEs se generan siempre de forma aleatoria y, para los APs, se asume que cada una dispone de una antena.

Los parámetros clave que definen un escenario son los siguientes:

- L: número total de APs.
- K: número total de UEs.
- N: número de antenas por AP (fijo en todos los experimentos).
- ASD_{φ}: dispersión de la energía en ángulos horizontales alrededor de un UE (fijo en todos los experimentos pero con impacto en las respuestas de canal de los UEs).
- τ_p: número de pilotos ortogonales usados para estimación de canal (fijo en todos los experimentos ya que su impacto es determinante pero en la fase de estimación de canal).

A lo largo de los experimentos se emplean diferentes configuraciones para evaluar la escalabilidad y robustez de los modelos. Estas configuraciones se consiguen modificando los parámetros mencionados, que pueden tomar los valores mostrados en la Tabla 5.1.

Parámetro	**Valor**
Área en m^2	[500, 750, 1000, 1250, 1500]
Distribución de los APs	Aleatoria/Ordenada (rejilla)
L	$l \in [31, 75]$[1]
K	$k \in [5, 45]$
N	1
ASD_{varphi}	$10(\pi/180)$
τ_p	4

Tabla 5.1: Valores de los parámetros de los escenarios.

Esta diversidad permite observar el comportamiento de los algoritmos tanto en sistemas densos como en otros más dispersos. De este modo, se consigue una base sólida para el análisis de resultados.

[1] Verdaderamente depende del parámetro K. Si $K < 30$, el valor mínimo de L será 35, mientras que si $K >= 30$, L podrá tomar valores a partir de $K + 1$.

5.2 Simulación de escenarios

Para generar estos escenarios simulados, se adaptó la implementación del repositorio público del proyecto *Cell-Free Massive MIMO Book* [47].

A continuación, se describe el proceso de generación de escenarios, que se divide en dos etapas principales:

- **Simulación del entorno físico:** adaptación de la generación de simulaciones espaciales encontradas en [47].
- **Conversión del entorno en dataset:** paso de los escenarios simulados a formato tabular, apto para ser usado como entrada de los modelos.

5.2.1 Simulación del entorno físico

Esta es la parte inicial, que consiste en construir el entorno completo de simulación. El propósito principal es generar, de manera realista, las condiciones físicas que afectan a cómo se transmite la información entre los APs y los UEs, incluyendo factores como la distancia, el ruido, las interferencias y las características del canal de propagación. El resultado de este proceso es una matriz de ganancias de canal $\beta_{i,k}$, que recoge la intensidad esperada de la señal entre cada par AP-UE.

Para ello, se comienza definiendo los límites espaciales del escenario, generando una cuadrícula de la extensión deseada, y estableciendo las coordenadas para los UEs y APs según los parámetros de entrada. La posición de cada elemento se guarda como un número complejo, donde la parte real representa la coordenada horizontal (x) y la imaginaria, la vertical (y), lo que permite trabajar con ellas de modo más simple. A continuación, se calcula la distancia entre cada par AP-UE de la siguiente forma:

$$dist_{i,k} = \sqrt{(distVertical)^2 + (x_i - x_k)^2 + (y_i - x_k)^2}, \tag{5.1}$$

donde $distVertical$ es la diferencia de altura entre los APs, que suelen estar en postes o techos, y los UEs, que están a nivel del suelo.[1] Las variables x_i e y_i representan las coordenadas del i-ésimo AP, mientras que x_k e y_k se corresponden con las coordenadas para el UE k.

Una vez obtenida la distancia, se puede calcular la pérdida de trayectoria (denominada *path-loss*), que modela la atenuación de la señal al propagarse: cuanto más lejos está el receptor,

[1] En este proyecto, se utiliza $distVertical = 10$ como valor fijo.

más débil será la señal recibida[2]. Esta pérdida se calcula como:

$$PL_{i,k} = \alpha \cdot \log_{10}(dist_{i,k}) - c, \tag{5.2}$$

donde:

- $\alpha = 36.7$ es el exponente de pérdida de trayectoria,
- $c = -30.5$ es un término constante.[3]

A esta pérdida, correspondiente al *pathloss*, se le añade una componente aleatoria que permite modelar el *shadowing* y que, por tanto, simula el desvanecimiento de la señal por objetos como edificios o árboles. Esta componente de *shadowing* se modela con una variable aleatoria gaussiana tal que:

$$S_{i,k} \sim \mathcal{N}(0, \sigma^2), \tag{5.3}$$

donde $\sigma = 4$dB, que representa un valor razonable para un entorno urbano.

Además, se considera también la presencia de ruido térmico, que afecta todas las transmisiones. La potencia de este ruido, en dBm, se calcula como:

$$\sigma^2_{ruido} = -174 + 10 n_p \log_{10}(B) + noiseFig \tag{5.4}$$

donde:

- $B = 20 \cdot 10^6$ es el ancho de banda del canal en Hz
- $n_p = 1$ es el número de portadoras consideradas en el sistema,[4]
- $noiseFig = 7$ es la figura de ruido del receptor.[3]

Seguidamente, con estos elementos se puede obtener la ganancia del canal, que indica la fuerza de la señal recibida, y se calcula como:

$$G^{(dB)}_{i,k} = -PL_{i,k} + S_{i,k} + \sigma^2_{ruido}. \tag{5.5}$$

[2] Todas las formulaciones utilizadas en este apartado se corresponden con formulaciones estándar descritas en obras como [48] y en recomendaciones del 3GPP [49], aunque fueron adaptadas a los parámetros del entorno simulado.

[3] Los valores de estos términos se corresponden a configuraciones comunes y estandarizadas para simulaciones de redes móviles en entornos urbanos. Han sido ampliamente usados en artículos y simulaciones académicas como [11, 48, 49].

[4] Las portadoras son señales que se usan para enviar información por el canal. Cada portadora puede transportar datos de forma independiente. En este caso se usa una por simplicidad.

Este valor se transforma a escala lineal (más cómodo para tratar como datos para los modelos) mediante la siguiente expresión:

$$\beta_{i,k} = G_{i,k}^{(lin)} = 10^{\frac{G^{(dB)}}{10}}. \tag{5.6}$$

Una vez obtenida esta ganancia para todos los pares AP-UE, se selecciona para cada UE un AP maestro, que se corresponde con aquel que proporciona la mejor ganancia de canal. Este maestro será clave para asignar la portadora piloto (τ_p), que es una señal corta que los UEs transmiten para identificarse ante la red.

El número de pilotos disponibles es limitado, por lo que los primeros UEs reciben un piloto único, y los siguientes tendrán que compartirlos en función del nivel de interferencia de cada uno.

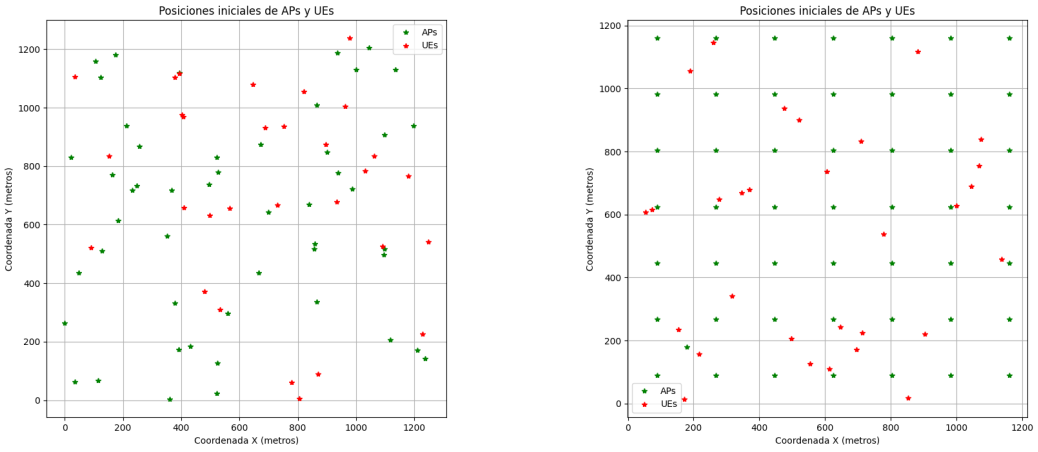
Todo este proceso se realiza en una función llamada `functionSetup()`, que tiene como parámetros de entrada los mencionados en la sección anterior: Área (m^2), distribución, L, K, N, τ_p y ASD_φ. En el código 5.1, se puede observar un resumen de dicha función, formado principalmente por las secciones modificadas para este proyecto.

```

# Posiciones de las APs y parámetros ya definidos

for k in range(K):
    # Asignación de posiciones aleatorias a los UEs y cálculo de distancias
    UEposition = (np.random.rand() + 1j*np.random.rand()) * squarelength
    distances[:, k] = np.sqrt(distanceVertical**2 + np.abs(APpositions - UEposition)**2)[:,0]

    # Cálculo del shadowing
    if k == 0:
        shadowing = sigma_sf * np.random.randn(L)
    else:
        ...  # Cálculo de shadowing correlacionado con otros UEs

    # Ca´culo del pathloss, ganancia del canal y paso a potencia
    for ap in range(L):
        pathloss_dB = alpha * np.log10(distances[ap, k])
        gainOverNoisedB[ap, k] = constantTerm - pathloss_dB + shadowing[ap] - noiseVariancedBm
        powgain[ap, k] = db2pow(gainOverNoisedB[ap, k])
```

```
20
21      # Selección de la AP master
22      master = np.argmax(gainOverNoisedB[:, k])
23
24      # Selección de los pilotos
25      if k < tau_p:
26          pilotIndex[k] = k
27      else:
28          pilotInterference = np.zeros(tau_p)
29          for t in range(tau_p):
30              interfering = pilotIndex[:k] == t
31              pilotInterference[t] =
    np.sum(db2pow(gainOverNoisedB[master, :k][interfering]))
32          pilotIndex[k] = np.argmin(pilotInterference)
```

Listing 5.1: Resumen de la generación de escenarios

Asimismo, en la Figura 5.1, podemos ver dos ejemplos de escenarios simulados, donde los dispositivo usuarios están representados en rojo y los punto de accesos, en verde. Ambos escenarios cuentan con un área de $1250m^2$, $K = 30$, $L = 50$ y distinta distribución de los APs. En la figura (a), los APs están situadas aleatoriamente y, en la (b), siguen una distribución ordenada en forma de rejilla.



(a) (b)

Figura 5.1: Simulaciones usando área $= 1250m^2, K = 30, L = 50$ y con las APs siguiendo una distribución aleatoria en (a) y ordenada en (b). *(Generación propia)*

5.2.2 Conversión en dataset

Una vez generados los datos correspondientes al escenario físico, se transforman en un *dataset* tabular, apto para funcionar como entrada de los modelos.

Antes de explicar cómo se realiza esto, hay que recordar el parámetro M, no mencionado en secciones anteriores de este capítulo por ser considerado una parte del procesamiento de los datos más que un elemento implícito en la generación de las simulaciones.

A cada UE se le asocia inicialmente un subconjunto reducido de APs, lo que simula una limitación realista de conectividad local. Para un UE i, se le asignan los M_k APs más próximos como candidatos a servirle, descartando automáticamente el resto. Esta selección, basada en la distancia geométrica, permite reducir el espacio de búsqueda sin comprometer significativamente el rendimiento.

Esta fase de preselección del subconjunto de APs candidatos también es clave a la hora de obtener las distintas configuraciones usadas para evaluar los modelos. De modo que, si se completa la Tabla 5.1, constaría de 5 parámetros variables y 3 fijos, como se ve en la Tabla actualizada 5.2.

Parámetro	Valor
Área en m^2	[500, 750, 1000, 1250, 1500]
Distribución de los APs	Aleatoria/Ordenada (rejilla)
L	$l \in [31, 75]$[1]
K	$k \in [5, 45]$
M	$[5, 10, 15, 20]$
N	1
ASD_φ	$10(\pi/180)$
τ_p	4

Tabla 5.2: Valores de los parámetros de los escenarios incorporando la fase de filtrado de APs.

Ahora sí, una vez generado el escenario físico, se transforma en un *dataset* tabular apto para

actuar como entrada de los modelos. Para cada UE k, se recorren los M_k APs que le han sido asignados como activos y, para cada uno de esos enlaces, se calculan y almacenan:

- Ambos identificadores (el del UE y el del AP).
- La ganancia del canal (normalizada entre [0,1]).
- La interferencia generada por el resto de UEs hacia ese AP (normalizada entre [0,1]).
- La distancia entre ambos dispositivos.

Además, para mantener una estructura homogénea en la tabla, se rellenan filas "nulas" con APs ficticias, asignándoles ganancias, interferencia y distancias iguales a 0 hasta llegar a M_{max}. Esto se hace para su uso posterior en los modelos con una configuración variable de M_k, ya que estos modelos requieren una cantidad fija de entradas por UE.

El resultado final es un `DataFrame` uniforme que contiene una fila por cada combinación UE-AP, y que puede usarse para entrenar los modelos, visualizar escenarios o evaluar el comportamiento de las redes bajo distintas configuraciones.

5.3 Métricas de evaluación

La evaluación de los distintos modelos implementados se llevó a cabo mediante un conjunto de métricas que abarcan tanto aspectos técnicos relacionados con la calidad de la asignación de recursos, como la eficiencia computacional. A continuación, se describen y justifican las principales métricas utilizadas en el proyecto.

5.3.1 Sum-rate del sistema

El *sum-rate*, descrito en capítulos anteriores, es la métrica principal empleada para evaluar la calidad de las asignaciones realizadas por los modelos. Esta métrica permite medir la eficiencia espectral de la red y refleja directamente el impacto de las decisiones de asignación en el rendimiento global del sistema. Por ello, su uso resulta especialmente relevante en escenarios de *Cell-Free Massive MIMO*, donde el objetivo es maximizar el aprovechamiento de los recursos del sistema de manera eficiente y conseguir la mayor eficiencia espectral posible (ya que eso supone mayor velocidad de transmisión).

5.3.2 Tiempo de ejecución

Para evaluar la viabilidad práctica de los modelos, se midió el tiempo medio necesario para generar una asignación (realizar una predicción) a partir de un escenario simulado. Esta

métrica permite identificar enfoques que presentan costes computacionales elevados ya que, aunque obtengan buenos resultados, esto podría comprometer su aplicabilidad en tiempo real.

La medición se llevó a cabo utilizando la librería `time` de Python, registrando los tiempos de ejecución en las mismas condiciones para todos los modelos. Esta métrica es habitual en evaluaciones comparativas de algoritmos en inteligencia artificial aplicada [50].

5.3.3 Uso de memoria

Además del tiempo de ejecución, se analizó el consumo de memoria durante el funcionamiento de cada modelo. Este aspecto es clave para determinar la escalabilidad de cada aproximación y el potencial para ser usada en entornos con recursos limitados.

El uso de memoria se registró mediante la librería `memory_profiler`, que permite monitorizar el consumo en tiempo real durante la ejecución del código. Esta información complementa al análisis de tiempo y permite valorar la eficiencia general de cada enfoque.

5.3.4 Estabilidad y convergencia

Como vimos en el capítulo 4, en el caso de los modelos basados en redes neuronales, se estudió la evolución de diferentes métricas durante las fases de entrenamiento. En concreto, se analizó la convergencia de la recompensa obtenida por episodio para el modelo basado en DCB y la estabilidad de las funciones de pérdida tanto para el basado en DCB, como para los basados en pérdida.

Estas observaciones permiten determinar si el modelo está aprendiendo de forma efectiva, si tiende a la sobreexploración o si sufre problemas de sobreajuste. Estas métricas son estándar en el análisis de entrenamiento de modelos de este tipo [15].

5.3.5 Análisis de sensibilidad a parámetros

Finalmente, se analizaron los efectos de modificar diversos parámetros del entorno de simulación sobre las métricas descritas. En concreto, se estudiaron variaciones en:

- El número de APs totales en el escenario L.
- El número de dispositivos UE K.
- El número de APs activos por UE M_k.
- El tamaño del área del escenario.

Estos análisis permiten valorar la robustez de cada modelo ante diferentes configuraciones del sistema, lo que resulta fundamental para validar su escalabilidad.

Aunque el simulador permite colocar los APs de forma aleatoria, durante las evaluaciones siempre se empleó la disposición ordenada. Esta decisión se debe a que una distribución ordenada permite evaluar los algoritmos de manera más controlada y reproducible. Además, ayuda a evitar los posibles sesgos de una ubicación aleatoria y hace más clara la interpretación de los resultados en condiciones uniformes de cobertura. Por último, este tipo de distribución representa una situación más realista para la fase de evaluación.

5.4 Análisis de resultados

A partir de los escenarios y las métricas descritas previamente, en esta sección se analiza el comportamiento de los distintos modelos implementados bajo múltiples configuraciones. El objetivo es entender su rendimiento en términos de eficiencia espectral, así como desde un punto de vista computacional.

Para facilitar la interpretación de los resultados obtenidos, se ha desarrollado un cuadro de mando interactivo en Power BI que permite comparar visualmente el funcionamiento de los modelos en función de los parámetros de configuración. A continuación, se presentan los principales hallazgos del análisis.

Finalmente, antes de entrar en los distintos análisis, conviene hacer una breve reflexión sobre las diferencias observadas entre las variantes de los modelos basados en pérdida. Ya se mencionó que se probaron dos variantes distintas para la función objetivo: una basada en la maximización directa del *rate* ($-R_k$), y otra basada en su inverso ($\frac{1}{R_k}$). En el Capítulo 4 se observó que la función inversa generaba una curva de pérdida más suave y estable, mientras que la función negativa tenía más oscilaciones. Esto se atribuye a que la función inversa penaliza más los valores pequeños del *rate*, forzando al modelo a evitar asignaciones malas, mientras que la negativa explora soluciones más agresivas que, aunque inestables, pueden ofrecer un mayor rendimiento medio.

5.4.1 Impacto de los parámetros del escenario

La evaluación de los modelos se llevó a cabo simulando su comportamiento sobre un conjunto de escenarios independientes, generados con distintas configuraciones de red. En concreto, para cada configuración específica de parámetros, se llevan a cabo 500 simulaciones

y, en cada una, se instancia un nuevo escenario sobre el que son evaluados en paralelo cada uno de los modelos, así como una asignación aleatoria de APs que funciona como *baseline* trivial. Este procedimiento asegura una comparación justa.

En el código 5.2, se puede ver un resumen del proceso de evaluación que, para cada episodio o simulación, consiste en los siguientes pasos:

1. Se genera un nuevo escenario simulado de acuerdo a los parámetros específicos considerados. Estos escenarios difieren básicamente en la disposición geométrica de los UEs y, como consecuencia, en las distancias respecto a los APs, las ganancias de canal y las interferencias correspondientes.

2. Se agrupan los datos necesarios para cada par UE-AP y se transforman en vectores de entrada adecuados para los modelos.

3. Cada modelo predice las acciones a partir de la información que hay en los datos de entrada.

4. Se calcula el *sum-rate* total por escenario para cada aproximación, y se almacena la media final.

```
num_episodes=500

# Evaluación de la predicción dle modelo
def binary_action(pred, threshold, Apss=None):
    binary_action = (pred > threshold).astype(int)
    if Apss is not None:
        binary_action[Apss:] = 0
    return binary_action

def process_model(state, pred, threshold, max_gain, Apss=None):
    action = binary_action(pred, threshold, Apss)
    return calculate_sum_rate(np.expand_dims(state.flatten(),
    axis=0), action, max_gain=max_gain)

# Inicialización de listas de resultados
random_sum_rates, DCB_sum_rates, CL_sum_rates, LBI_sum_rates,
    LBN_sum_rates = [], [], [], [], []

def evaluate_model(num_episodes, Ls=50, Ks=30, Apss=10,
    possible_actions=False):

```

```
    for episode in tqdm(range(num_episodes), desc="Evaluando modelo"):

        # Generación y preparación de los datos simulados
        ...

        # Predicción con los modelos
        states_batch = np.stack(agg_df['state'].values)
        DCB_preds = DCB.predict(states_batch, verbose=0)
        LBI_preds = LBI.predict(X_episode, verbose=0)
        LBN_preds = LBN.predict(X_episode, verbose=0)

        # Inicialización de los acumuladores del episodio
        random_sum_rate = DCB_sum_rate = LBI_sum_rate = LBN_sum_rate = 0

        for i, state in enumerate(pr_df['state']):

            state_input = np.array(state).reshape(1, -1)

            # Asignación aleatoria
            random_action = random.choice(possible_actions)
            random_sum_rate += calculate_sum_rate(state_input, random_action, max_gain=max_gain)

            # Asignación con los modelos DCB y loss-based
            DCB_sum_rate += process_model(state, DCB_preds[i], threshold=0.01, max_gain=max_gain, Apss=Apss)
            LBI_sum_rate += process_model(state, LBI_preds[i], threshold=0.5, max_gain=max_gain)
            LBN_sum_rate += process_model(state, LBN_preds[i], threshold=0.5, max_gain=max_gain)

        # Asignación con clustering difuso
        threshold = buscar_threshold_optimo(df, max_gain, Ks, Ls, Apss, noise=N0)
        CL_sum_rate, _ = fuzzy_clustering_aps(df, max_gain, Ks, Ls, Apss, N0, m=2, threshold=threshold)

        # Almacenamiento de resultados del episodio
        random_sum_rates.append(random_sum_rate)
        DCB_sum_rates.append(DCB_sum_rate)
        CL_sum_rates.append(CL_sum_rate)
        LBI_sum_rates.append(LBI_sum_rate)
        LBN_sum_rates.append(LBN_sum_rate)

```

```
    # Resultados finales
    results = {
        "Aleatorio": [np.mean(random_sum_rates)],
        "DCB": [np.mean(DCB_sum_rates)],
        "Clustering (Rate)": [np.mean(CL_sum_rates)],
        "Loss-driven (Inv)": [np.mean(LBI_sum_rates)],
        "Loss-driven (Neg)": [np.mean(LBN_sum_rates)]
    }

    return results
```

Listing 5.2: Código correspondiente a la evaluación de los modelos.

Tras llamar a la función de evaluación, los resultados con las distintas configuraciones se almacenan en un conjunto de ficheros `.csv` con el rendimiento promedio de cada modelo en los distintos escenarios, lo que proporciona una base sólida para el análisis.

Una vez detallado el proceso para obtener los resultados, se procede a mostrar los resultados obtenidos con su correspondiente análisis. La estructura de los diferentes análisis consistirá en una descripción general de los escenarios de evaluación (parámetros fijos y valores del parámetro evaluado), la descripción de los resultados observados y la interpretación de los mismos.

Número de APs candidatos (M)

Descripción general de los escenarios:

Para este análisis se fijaron los siguientes parámetros:

- $K = 30$ UEs.
- $L = 50$ APs en el escenario.
- Área $= 1000m^2$.
- Distribución ordenada (en rejilla) de los APs.

En este caso, se evaluaron los modelos tomando los valores $M \in [5, 10, 15, 20]$, que determinan el número de APs candidatos considerados tras el filtrado inicial.

Resultados observados:

Como se puede apreciar en la Figura 5.2, todos los modelos muestran una tendencia creciente en el *sum-rate* al incrementar el número de APs candidatos por UE, aunque con diferente intensidad:

- El modelo *Loss-driven* negativo alcanza los mejores resultados en todos los casos ($M \in [5, 10, 15, 20]$), manteniendo una mejora consistente sobre el resto de enfoques. Es conveniente recordar que este modelo se basa en minimizar directamente una función de pérdida basada en el negativo del valor de *sum-rate*.
- Los modelos *Clustering* y *Loss-driven* inverso –función de pérdida inversa del *sum-rate*– muestran también una mayor tendencia de subida al aumentar M que el modelo DCB, aunque el segundo no llega a alcanzar los valores de este.
- La asignación Aleatoria es la que menos se beneficia del incremento en el número de APs candidatos.

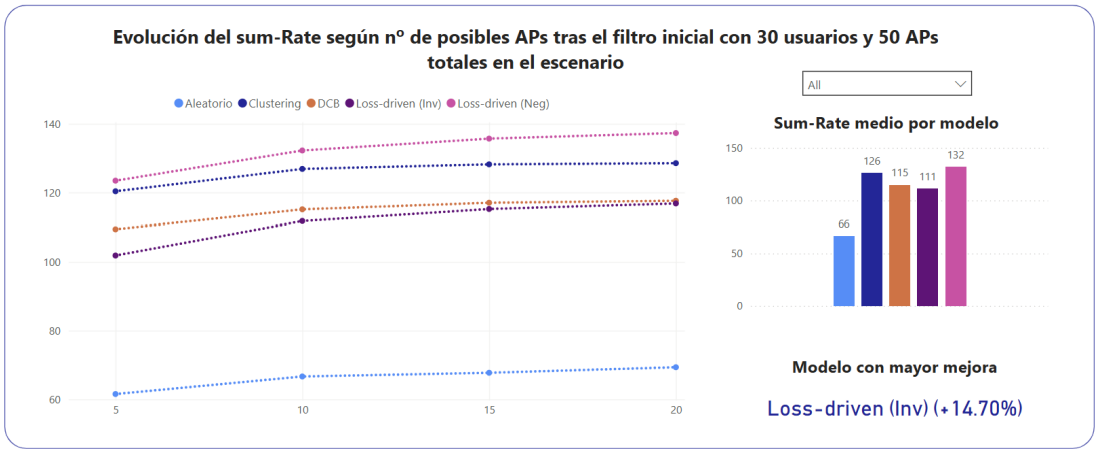


Figura 5.2: Resultados de la evaluación del parámetro M. *(Generación propia)*

Interpretación de los resultados:

La Figura 5.2 muestra cómo el aumento de M permite a los modelos acceder a un conjunto más amplio de posibles asignaciones, lo que incrementa las probabilidades de encontrar combinaciones favorables de APs. Los modelos más orientados a optimización directa como los *loss-based* o el de *clustering* consiguen explotar mejor esta mayor flexibilidad, mientras que el modelo DCB mantiene un crecimiento más moderado. Una posible explicación es que, al incorporar contexto, el modelo prioriza configuraciones estables y conservadoras que son robustas en distintas situaciones. Sin embargo, esto lleva a una menor explotación del espacio adicional cuando M crece y, por tanto, un menor aprovechamiento de las nuevas combinaciones disponibles.

Número de UEs (K)

Es importante mencionar que, aunque en la mayoría de los análisis se utiliza el *sum-rate* medio por escenario como métrica de comparación, en el caso concreto de variar el número de UEs (K), se emplea el *rate medio por UE*. Esto se debe a que un aumento en el número total de UEs incrementará el *sum-rate* global simplemente por acumulación, pero esto no implica una mejora real en el servicio. En este caso, se quiere observar cómo afecta la competencia entre UEs sobre un número limitado de recursos, por lo que se ha decidido utilizar el *rate* medio por UE como métrica a valorar.

Descripción general de los escenarios:

Para este análisis se fijaron los siguientes parámetros:

- $M = 10$ APs candidatos por UE.
- $L = 50$ APs en el escenario.
- Área $= 1000m^2$.
- Distribución ordenada de los APs.

En este caso, se evaluaron los modelos tomando valores de $K \in [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]$ UEs.

Resultados observados:

Como se comentó anteriormente, en la Figura 5.3 se muestra la evolución del *rate* medio por UE, lo que permite comparar de forma justa el impacto del crecimiento de UEs bajo un número de APs constante. Con respecto a los resultados obtenidos, se pueden destacar los siguientes puntos:

- Como era esperable, todos los modelos muestran una disminución en el *rate* medio por UE a medida que aumenta K debido al mayor reparto de recursos por UE. Esto implica una mayor posibilidad de interferencia, lo que supone un desafío importante en la asignación de APs para equilibrar la ganancia de asignar un nuevo AP frente a la interferencia que ello genera en los demás UEs conectados a ese AP.
- El modelo DCB muestra una de las pérdidas más bajas, lo que demuestra su capacidad de adaptación en escenarios más congestionados.
- Ambos modelos *Loss-driven*, así como el *clustering* difuso, siguen mostrando buen rendimiento, aunque tienen caídas más pronunciadas al aumentar la densidad de UEs.

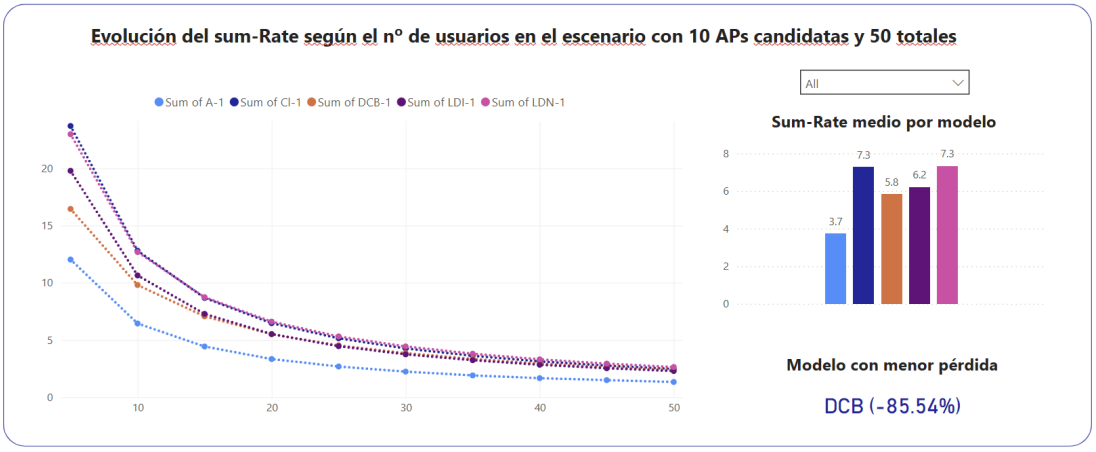


Figura 5.3: Resultados de la evaluación para distintos valores de K. *(Generación propia)*

Interpretación de los resultados:
Como se observa en la Figura 5.3, el crecimiento de K crea más competencia entre los UEs por los recursos de red disponibles. Mientras que los enfoques basados en *clustering* y pérdida aplican asignaciones de recursos basadas en criterios generales, el modelo DCB presenta una mejor capacidad de adaptación dinámica a los contextos particulares de cada UE. Esto le permite minimizar mejor la pérdida relativa conforme el escenario se vuelve más denso, lo que demuestra su mayor robustez en escenarios congestionados.

Número de APs totales en el escenario (L)

Descripción general de los escenarios:
Para este análisis se fijaron los siguientes parámetros:

- $M = 10$ APs candidatos por UE.
- $K = 30$ UEs.
- Área $= 1000m^2$.
- Distribución ordenada de los APs.

En este caso, se evaluaron los modelos tomando los valores de $L \in [35, 90] \; \forall \; l/5 = 0$.

Resultados observados:
En la Figura 5.4, se presentan los resultados obtenidos. A partir de estos resultados, se pueden extraer las siguientes conclusiones:

- Como era esperable, todos los modelos presentan una tendencia positiva a medida que L aumenta.
- La asignación basada en *Clustering* difuso es la que más ventaja obtiene al incrementar L. Sin embargo, su rendimiento se mantiene por debajo del modelo basado en pérdida con la función negativa del *sum-rate* para todo el rango de valores de L.
- Los modelos DCB y *Loss-driven* inverso mantienen trayectorias muy similares, creciendo con el aumento de L, pero con ganancias relativas inferiores.

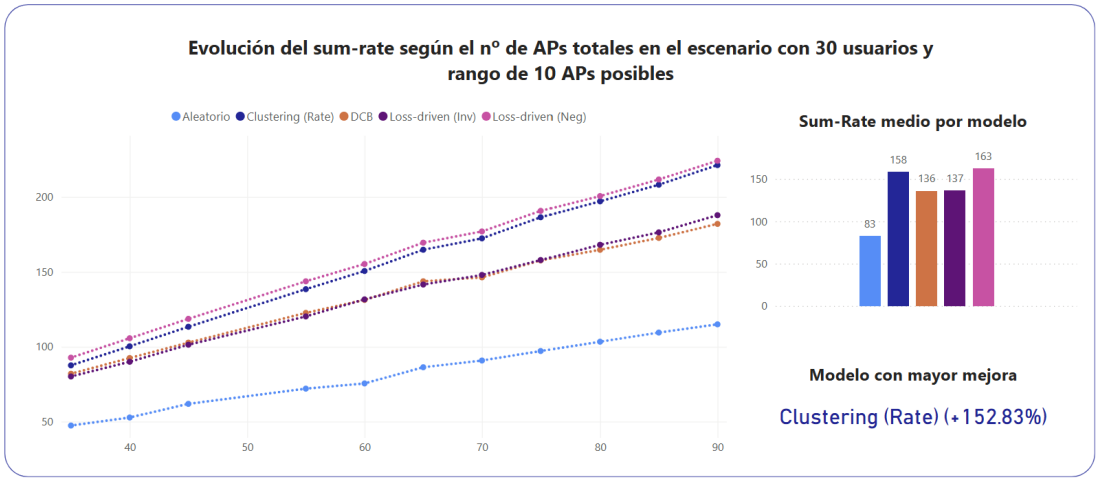


Figura 5.4: Resultados de la evaluación para distintos valores de L. *(Generación propia)*

Interpretación de los resultados:
En este caso ocurre algo parecido a lo que pasaba a la hora de aumentar M. Se puede ver en la figura 5.4 que el aumento en el número total de APs genera una red de acceso más densa, disminuyendo las distancias entre los UEs y los APs disponibles, lo que mejora las condiciones del canal. Además, aumentar el número de APs supone incrementar los grados de libertad del sistema para cancelar las interferencias causadas entre UEs. Estos efectos favorecen especialmente a aquellos modelos que no requieren entrenamientos complejos, ya que se benefician directamente de la mayor disponibilidad de enlaces y grados de libertad del sistema.

Sin embargo, modelos como el DCB siguen mejorando su rendimiento, pero de forma más gradual, ya que sus decisiones dependen del contexto completo. Esto los hace más robustos frente a situaciones complejas, pero algo más conservadores cuando los recursos disponibles son abundantes y no suponen un factor claramente limitante.

Área del escenario

Descripción general de los escenarios:

Para este análisis se fijaron los siguientes parámetros:

- $M = 10$ APs candidatos por UE.
- $K = 30$ UEs.
- $L = 50$ APs en el escenario.
- Distribución ordenada de los APs.

En este caso, se evaluaron los diferentes modelos tomando como valores para el tamaño del escenario los del conjunto Área $\in [500, 750, 1000, 1250, 1500]$ m^2.

Resultados observados:

La Figura 5.5 muestra una tendencia general de incremento en el *sum-rate* medio a medida que aumenta el área de cobertura. En concreto, se puede destacar que:

- Todos los modelos experimentan ligeras mejoras al incrementarse el tamaño área del área de cobertura.
- El modelo basado en DCB presenta en este caso la mayor ganancia relativa.

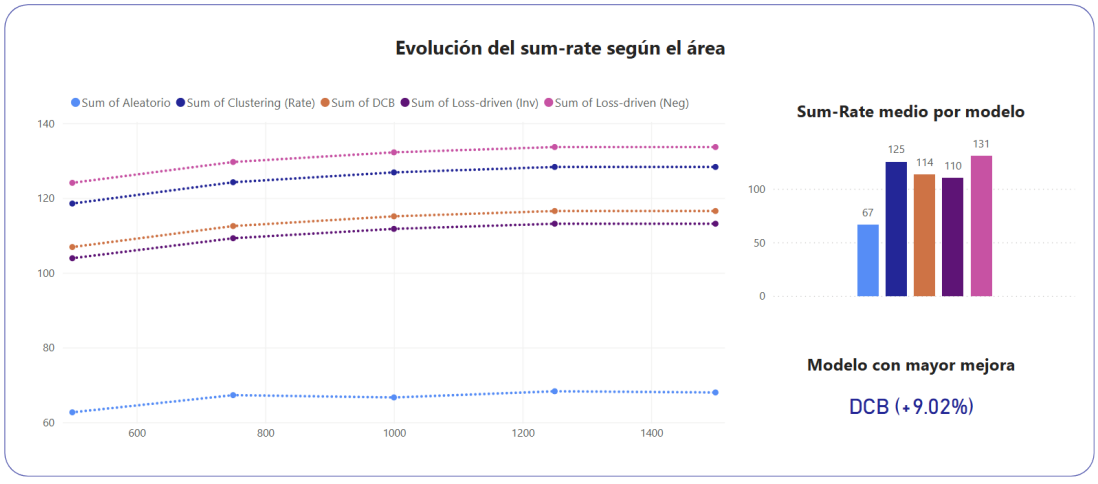


Figura 5.5: Resultados de la evaluación para distintos valores del área. *(Generación propia)*

Interpretación de los resultados:

Aunque, a priori, podría resultar contradictorio, se observa en la Figura 5.5 que, al ampliar

el área total, mejoran las ganancias proporcionadas por los modelos en este escenario. Esto ocurre porque no solo aumenta la separación media entre UEs y APs, lo que causaría un empeoramiento de la señal al aumentar la pérdida asociada al *pathloss*[5], sino que también se reduce la densidad de UEs por unidad de superficie. Esto último conlleva menores niveles de interferencia procedente de otros dispositivos, permitiendo a los algoritmos explotar más eficazmente la menor competencia.

En este contexto, el modelo DCB, al haber sufrido un entrenamiento más complejo teniendo en cuenta el contexto, muestra mayor capacidad y flexibilidad para adaptarse a esta mejora en los niveles de interferencia, optimizando así mejor sus asignaciones locales de APs en entornos más dispersos.

5.5 Análisis de eficiencia

En esta sección se evalúa la eficiencia computacional de los distintos modelos implementados. Para ello, se analizan dos métricas principales: el tiempo de ejecución, en segundos, que refleja la velocidad con la que cada modelo genera una asignación de APs, y el consumo de memoria, en GB, que refleja los recursos necesarios para llevar a cabo el proceso. Estas métricas permiten comparar la viabilidad práctica de cada enfoque, especialmente en escenarios con alta densidad de UEs o de APs.

Descripción general de los escenarios:
Para el análisis de eficiencia se reutilizaron los mismos escenarios generados para las evaluaciones de rendimiento anteriores, excluyendo las variaciones respecto al área de cobertura. Esto se debe a que los algoritmos operan sobre el conjunto de APs y UEs disponibles en cada escenario, sin depender directamente de la distribución espacial. Por tanto, la complejidad computacional y los consumos de recursos quedan principalmente ligados al tamaño de la entrada (M, K, L).

Resultados observados:
Los resultados obtenidos para este análisis se presentan en la Figura **??**. Entre los principales puntos a destacar, se puede comentar que:

- Los modelos basados en *Clustering* difuso son los más eficientes en consumo de memoria (prácticamente nulo) y tiempo de ejecución, destacando especialmente en los experimentos donde se varían K y M.

[5] Recordar que esta pérdida está asociada directamente con la propagación de las ondas electromagnéticas y aumenta cuadráticamente con la distancia.

- Los modelos basados en DCB y *Loss-driven* (en sus dos variantes) presentan tiempos de ejecución muy próximos entre sí, alrededor del rango (0.11, 0.13) segundos, con consumos de memoria igualmente bajos.
- El modelo aleatorio tiene tiempos de cómputo muy superiores, así como consumos notablemente mayores de memoria.

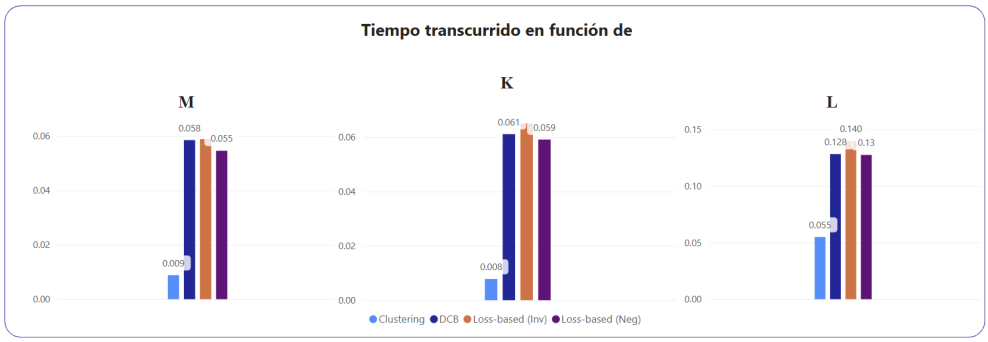


Figura 5.6: Gráfica de resultados de la evaluación de tiempo de ejecución. *(Generación propia)*

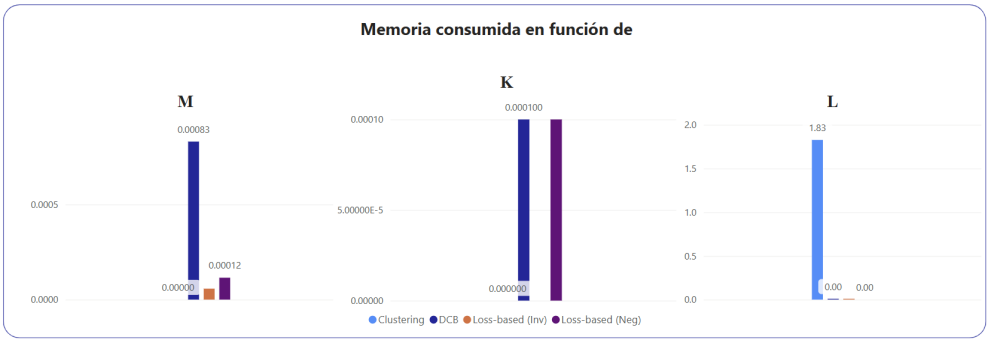


Figura 5.7: Gráfica de resultados de la evaluación de memoria consumida. *(Generación propia)*

Interpretación de los resultados:
Los resultados vistos en las Figuras 5.6 y 5.7 muestran que, en términos de eficiencia, el algoritmo de *Clustering* difuso destaca por su bajo consumo de tiempo y memoria, lo que lo hace muy adecuado para situaciones con recursos computacionales limitados o cuando se necesita tomar decisiones casi de inmediato. Por otro lado, los modelos de aprendizaje requieren más recursos durante el entrenamiento, pero durante la fase de inferencia consumen poca memoria y tiempo, lo que les permite funcionar bien incluso con muchos UEs o con dimensiones relativamente altas para el espacio de búsqueda.

Desde el punto de vista de la escalabilidad, los modelos DCB y basados en pérdida tienen una gran ventaja. Como trabajan sobre redes preentrenadas, el coste de operación se mantiene casi constante. Sin embargo, el *Clustering* difuso funciona bien en situaciones con pocos o un número moderado de UEs, pero podría tener problemas si el número de UEs o APs crece mucho, como se ve en la Figura **??**, en el caso del aumento del consumo de memoria cuando se incrementa el parámetro L. Un detalle a mencionar es que el modelo DCB usa un poco más de memoria que los basados en pérdida. Esto se debe a la estrategia de inferencia por lotes y al almacenamiento temporal de vectores de acción durante el cálculo del *sum-rate*. Sin embargo, este incremento es despreciable a escala práctica.

Quizás llama la atención la falta de la aproximación de asignación Aleatoria en las gráficas de las Figuras 5.6 y 5.7. Esto se debe a que sus valores de tiempo y memoria son significativamente superiores, lo que comprimía la escala de representación e impedía apreciar las diferencias entre las aproximaciones principales. Esto ocurre porque, aunque el modelo aleatorio es conceptualmente simple, su implementación práctica implica la generación y evaluación de un gran número de combinaciones posibles de activación de APs, lo que supone un coste computacional elevado. Por este motivo, sus resultados se muestran en las Figuras 5.8 y 5.9, donde se pueden visualizar sin distorsionar la escala del resto de aproximaciones.

Tiempo transcurrido en función de

Métrica	Aleatorio	Clustering	DCB	Loss-based (Inv)	Loss-based (Neg)
L	0.16	0.05500	0.12800	0.14000	0.12700
K	0.25	0.01600	0.12200	0.13000	0.11800
M	2.14	0.01800	0.11600	0.11800	0.10900
Total	**2.54**	**0.08900**	**0.36600**	**0.38800**	**0.35400**

Figura 5.8: Tabla de resultados de la evaluación de tiempo de ejecución. *(Generación propia)*

Memoria consumida en función de

Métrica	Aleatorio	Clustering	DCB	Loss-based (Inv)	Loss-based (Neg)
M	12.40	0.00000	0.00083	0.00006	0.00012
K	1.51	0.00000	0.00010	0.00000	0.00010
L	0.72	1.82700	0.00010	0.00005	0.00000
Total	**14.62**	**1.82700**	**0.00103**	**0.00011**	**0.00022**

Figura 5.9: Tabla de resultados de la evaluación de memoria consumida. *(Generación propia)*

Capítulo 6

Conclusiones y trabajo futuro

El presente proyecto ha tenido como objetivo abordar el problema de la asignación de APs en entornos de comunicaciones Cell-Free Massive MIMO desde una perspectiva *user-centric*, explorando diferentes aproximaciones basadas en técnicas de aprendizaje automático y modelización matemática. A lo largo del desarrollo, se han diseñado y evaluado distintos modelos, desde heurísticas tradicionales hasta propuestas basadas en aprendizaje por refuerzo contextual (RL), permitiendo analizar el comportamiento de cada aproximación tanto a nivel de rendimiento como de eficiencia computacional.

En este capítulo se presentan las principales conclusiones del trabajo realizado, así como posibles líneas de mejora y continuación que podrían desarrollarse en el futuro para seguir optimizando el sistema y abordar nuevos retos derivados de su aplicación.

6.1 Conlusiones

Desde un punto de vista personal, este proyecto ha supuesto una gran oportunidad para adentrarme en el diseño de sistemas de comunicación avanzados, adentrándome en un área innovadora como las redes *Cell-Free Massive MIMO*, onsideradas una de las tecnologías clave en el desarrollo de las próximas generaciones de redes móviles. A lo largo del trabajo, he podido adquirir un conocimiento más sólido sobre aspectos fundamentales como la propagación de señales, la interferencia y la asignación eficiente de recursos en entornos inalámbricos. Al mismo tiempo, he desarrollado habilidades prácticas en la creación de simuladores y en la implementación de modelos de aprendizaje automático. Además, me ha permitido integrar de forma práctica el procesamiento eficiente de datos complejos con el diseño y entrenamiento de modelos, reforzando así la conexión entre teoría y aplicación en un contexto tecnológico actual y desafiante.

En cuanto a las conclusiones técnicas, durante el desarrollo de este proyecto se abordó el problema de la asignación de APs en entornos Cell-Free Massive MIMO, proponiendo varios enfoques basados en aprendizaje automático para optimizar las decisiones de asignación en función del contexto de cada UE. Para ello, se diseñó e implementó un sistema completo de simulación capaz de generar escenarios realistas, modelando la distribución de los UEs, los parámetros de canal, la configuración de los APs y UEs, y las condiciones de interferencia.

Sobre esta base, se desarrolló un modelo de aprendizaje basado en DCB, capaz de tomar decisiones de asignación en función de las características locales de cada UE, considerando tanto la ganancia de la señal directa como la interferencia procedente del resto de UEs presentes en el sistema. Este enfoque permite simplificar el problema en comparación con técnicas clásicas de aprendizaje por refuerzo secuencial, manteniendo un buen equilibrio entre complejidad y rendimiento.

Para evaluar el comportamiento del modelo propuesto, se diseñaron y compararon además distintas aproximaciones adicionales, incluyendo un modelo aleatorio de referencia, un modelo heurístico basado en *clustering difuso* y dos variantes de modelos basados en pérdida optimizados mediante funciones de coste basadas en el *sum-rate* inverso y negativo, respectivamente. Los resultados experimentales mostraron que el modelo basado en DCB ofrece un rendimiento competitivo frente a estas aproximaciones, especialmente en escenarios donde existe una alta competencia entre UEs por los recursos disponibles, gracias a su capacidad para adaptarse dinámicamente al contexto. En cambio, los modelos basados en pérdida, particularmente el que emplea la función negativa del *sum-rate*, alcanzan los mejores valores medios de rendimiento en escenarios con mayor disponibilidad de recursos, aunque existe el riesgo de que su desempeño empeore si se introduce dinamismo o complejidad al entorno. Por su parte, el modelo de *clustering*, tiene un bajo coste computacional en la mayoría de los casos, y consigue resultados razonables sin necesidad de entrenamiento. Sin embargo, esta superioridad en eficiencia se disipa con escenarios más saturados

Además del análisis de rendimiento, se llevó a cabo un estudio detallado de eficiencia computacional, evaluando tanto los tiempos de inferencia como el consumo de memoria de cada modelo. Por una parte, este análisis confirmó la viabilidad operativa de los modelos de aprendizaje propuestos, que mantienen un coste computacional bajo y estable durante la inferencia, permitiendo su aplicación en sistemas de comunicación de gran escala. Por otra, permitió identificar limitaciones y posibles áreas de mejora en los modelos evaluados, tanto a nivel de formulación del problema como en la capacidad de generalización ante escenarios más complejos o dinámicos, lo que abre la puerta a futuras líneas de investigación.

6.2 Trabajo futuro

Aunque los resultados obtenidos son prometedores, existen varias líneas de trabajo que podrían explorarse para seguir ampliando y profundizando el sistema desarrollado. A continuación, se presentan algunas propuestas de trabajo futuro orientadas a extender el enfoque actual hacia una optimización más general en la asignación de recursos en sistemas *Cell-Free Massive MIMO*:

- **Introducción de dinamismo en los escenarios de simulación**: Incorporar variabilidad temporal, como modelos de desplazamientos de los UEs, fluctuaciones en las condiciones de canal o cambios de carga en los APs, permitiría evaluar la capacidad de adaptación de los modelos en escenarios realistas. Este enfoque permitiría, además, comprobar si los modelos basados en contexto (como el DCB) presentan una ventaja adaptativa frente a aproximaciones más estáticas, como son los modelos basados en pérdida, cuya capacidad de ajuste podría verse limitada en entornos variables.

- **Extensión de las funciones de recompensa**: Ampliar las métricas optimizadas por los modelos, incorporando factores como el consumo energético de los APs, evitar *rates* individuales de UEs muy dispares, restricciones de calidad de servicio individuales, un balance global de carga, la estabilidad de las asignaciones a lo largo del tiempo o la penalización de transiciones excesivas entre configuraciones, permitiría adaptar el sistema a escenarios operativos más complejos y realistas, donde múltiples objetivos deben ser considerados simultáneamente.

- **Validación sobre datos reales o simuladores estandarizados**: Aplicar los modelos desarrollados sobre datos provenientes de despliegues reales, o en entornos de simulación estandarizados como *Open RAN*[1], permitiría validar el comportamiento de los algoritmos en condiciones realistas y no controladas.

- **Evaluación de robustez frente a ruido en los parámetros de entrada**: Analizar cómo se comportan los modelos ante incertidumbres o errores de medición en los parámetros de ganancia e interferencia. Esto permitiría aproximar mejor el sistema a condiciones reales, donde la información disponible sobre el canal o los enlaces es parcial o imperfecta.

[1] *Open RAN* es un marco abierto para el desarrollo de redes de acceso radio, que permite evaluar algoritmos de asignación de recursos en escenarios reproducibles y comparables entre diferentes investigaciones [51].

Estas propuestas permitirían consolidar y extender el enfoque desarrollado en este trabajo, acercándolo paulatinamente a una formulación más completa del problema global de *scheduling* en sistemas Cell-Free Massive MIMO. Esto integraría no solo la asignación de APs, sino también otras dimensiones como la planificación temporal de recursos, el control de potencias de transmisión y la coordinación multiusuario, permitiendo así abordar de forma adaptativa la optimización completa de los recursos disponibles en la red.

De esta forma, el trabajo desarrollado proporciona una base sólida sobre la que continuar explorando y depurando esta línea de investigación, contribuyendo al desarrollo de soluciones más eficientes y adaptativas para la asignación de recursos en redes de comunicaciones inalámbricas.

Lista de acrónimos

AP punto de acceso. iv, vi, 2, 3, 5, 6, 9–13, 17–25, 31, 34–39, 41, 42, 44, 46, 47, 49, 50, 52–57, 59–65, 67–73, 75–79

BS estación base. 9–12

CB Contextual Bandits. 15, 16

CPUs Unidades de procesamiento centralizado. iv, 11

DCB Deep Contextual Bandits. 2–7, 9, 16–19, 22, 24, 25, 34, 35, 39, 42, 46, 47, 52–55, 63, 68–75, 77, 78

DQN *Deep Q-Network*. 24

DRL Deep Reinforcement Learning. 24

FCM *Fuzzy C-Means*. 20

GNNs redes neuronales basadas en grafos. 24

MDP proceso de decisión de Markov. 14, 15

MIMO Multiple-Input Multiple-Output. 1, 5, 9–12, 17, 19, 24, 34, 35, 52, 55, 57, 62, 76–79

QoS calidad del servicio. 10–12

RL Reinforcement Learning. 13–15, 18, 19, 46, 76

SINR relación señal-interferencia-más-ruido. 19, 50

UE dispositivo usuario. iv, vi, 1–6, 9–13, 17, 18, 20–27, 32, 34–39, 41, 42, 44, 47, 49, 50, 53–57, 59–63, 65, 67, 69–75, 77, 78

VS Code *Visual Studio Code*. 30

Bibliografía

[1] A. Samad, "1g, 2g, 3g, 4g and 5g wireless phone technology explained – meaning and differences," 2022. [En línea]. Disponible en: https://www.linkedin.com/pulse/1g-2g-3g-4g-5g-wireless-phone-technology-explained-meaning-samad/

[2] B. . S. Demir, *Foundations of User-Centric Cell-Free Massive MIMO*, 2022.

[3] Piscine26, "Machine learning for unsupervised learning supervised learning reinforcement learning." [En línea]. Disponible en: https://www.freepik.com/premium-vector/machine-learning-unsupervised-learning-supervised-learning-reinforcement-learning_280220708.htm

[4] E. K. Jacob Murel, "What is reinforcement learning?" 2024. [En línea]. Disponible en: https://www.ibm.com/think/topics/reinforcement-learning

[5] G. Fei, "Contextual bandit for marketing treatment optimization," 2021. [En línea]. Disponible en: https://www.aboutwayfair.com/careers/tech-blog/contextual-bandit-for-marketing-treatment-optimization

[6] P. Ágiles, "Desarrollo iterativo e incremental." [En línea]. Disponible en: https://proyectosagiles.org/desarrollo-iterativo-incremental/

[7] [En línea]. Disponible en: https://martabenavente.atlassian.net/jira/software/projects/KAN/boards/1

[8] 2025. [En línea]. Disponible en: https://es.talent.com/salary?job=ingeniero+de+datos&utm_source=chatgpt.com

[9] H. H. S. S. O. S. David Gesbert, Stephen Hanly and W. Yu, "Multi-cell mimo cooperative networks: A new look at interference," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 2010.

[10] M. F. Sirichai Hemrungrote1, Toshikazu Hori and K. Nishimori, "Effects of path visibility on urban mimo systems," *ECTI TRANSACTIONS ON COMPUTER AND INFORMATION TECHNOLOGY*, 2011.

[11] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson, and T. L. Marzetta, "Cell-free massive mimo versus small cells," *IEEE Transactions on Wireless Communications*, 2017.

[12] H. Q. Ngo, "Cell-free massive mimo," *Encyclopedia of Wireless Networks*, 2020.

[13] A. Pantelidou and A. Ephremides, "The scheduling problem in wireless networks," *JOURNAL OF COMMUNICATIONS AND NETWORKS*, 2009.

[14] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive mimo networks: Spectral, energy, and hardware efficiency," *Foundations and Trends in Signal Processing*, 2017.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 2018.

[16] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, 1996.

[17] A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. E. Schapire, "Taming the monster: A fast and simple algorithm for contextual bandits," in *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[18] A. Bietti, A. Agarwal, and J. Langford, "A contextual bandit bake-off," *Journal of Machine Learning Research*, 2021.

[19] C. Riquelme, G. Tucker, and J. Snoek, "Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling," *arXiv preprint arXiv:1802.09127*, 2018.

[20] S. Agrawal and N. Goyal, "Thompson sampling for contextual bandits with linear payoffs," *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[21] T. L. Marzetta, "Noncooperative cellular wireless with unlimited numbers of base station antennas," *IEEE Transactions on Wireless Communications*, 2010.

[22] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms.* Springer Science & Business Media, 1981.

[23] K.P.Murphy, *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[24] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson, and T. L. Marzetta, "Cell-free massive mimo: Uniformly great service for everyone," *IEEE*, 2017.

[25] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive mimo networks: Spectral, energy, and hardware efficiency," *Foundations and Trends in Signal Processing*, 2017.

[26] S. Buzzi, C. D'Andrea, and T. L. Marzetta, "User-centric 5g cellular networks: Resource allocation and comparison with the cell-free massive mimo approach," *IEEE*, 2017.

[27] F. Liang, C. Zhong, and et al., "Deep reinforcement learning for resource allocation in wireless networks," *IEEE*, 2019.

[28] U. Challita, W. Saad, and C. Bettstetter, "Machine learning for wireless connectivity and security in the internet of things," *IEEE*, 2019.

[29] O. F. D. P.-A. L. C. Dariel Pereira-Ruisánchez, Michael Joham and W. Utschick, "Access point assignment for cell-free massive mimo networks using graph neural networks," *IEEE*, 2025.

[30] ASUS, "Asus vivobook pro 15 oled," 2024. [En línea]. Disponible en: https://www.asus.com/es/laptops/for-home/vivobook/asus-vivobook-pro-15-oled-n6506/techspec/

[31] P. S. Foundation, "Python language reference, version 3.10," 2023. [En línea]. Disponible en: https://www.python.org

[32] J. Wang, "Visual studio code: A powerful ide for ai development," *IEEE*, 2020.

[33] C. R. Harris, K. J. Millman, S. J. van der Walt, and et al., "Array programming with numpy," *Nature*, 2020.

[34] W. McKinney, "Data structures for statistical computing in python," 2010.

[35] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, 2020.

[36] J. D. Hunter, "Matplotlib: A 2d graphics environment," *IEEE*, 2007.

[37] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016.

[38] J. S. Warner, E. J. Gorlin *et al.*, 2019. [En línea]. Disponible en: https://scikit-fuzzy.github.io

[39] S. Chacon and B. Straub, *Pro Git.* Apress, 2014.

[40] F. Mittelbach and M. Goossens, *The LaTeX Companion.* Addison-Wesley, 2004.

[41] Atlassian, 2023. [En línea]. Disponible en: https://www.atlassian.com/software/jira/guides

[42] M. Corporation, 2023. [En línea]. Disponible en: https://learn.microsoft.com/en-us/power-bi/

[43] T. M. Cover and J. A. Thomas, "Elements of information theory," *Wiley-Interscience*, 2006.

[44] G. H. Golub and C. F. V. Loan, "Matrix computations," *Johns Hopkins University Press*, 2013.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[46] Q. Mao, O. Simeone, and Y. Niv, "Deep learning for wireless communication systems: An overview," *IEEE Access*, 2017.

[47] Björnson, 2021. [En línea]. Disponible en: https://github.com/emilbjornson/cell-free-book/tree/main/code

[48] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[49] "Study on channel model for frequencies from 0.5 to 100 GHz," Tech. Rep., 2019. [En línea]. Disponible en: https://www.3gpp.org/ftp/Specs/archive/38_series/38.901/

[50] S. Raschka and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing, 2020.

[51] O-RAN Alliance, "O-ran: Towards an open and smart ran," 2022. [En línea]. Disponible en: https://www.o-ran.org