Marco de Desarrollo Software e Implementación de Algoritmos de Inteligencia Artificial para la Gestión de Redes Radio 5G

I. Vilà, O. Sallent, J. Pérez-Romero
irene.vila.munoz@upc.edu, sallent@tsc.upc.edu, jordi.perez-romero@upc.edu
Dept. de Teoría de la Señal y Comunicaciones, Universitat Politècnica de Catalunya (UPC), Barcelona

***Abstract*—The increase in complexity of 5G and beyond mobile communications networks to accommodate multiple services with stringent requirements has led to the introduction of Artificial Intelligence (AI) capabilities for automating their management and operation, particularly in the Radio Access Network (RAN). Although there exist a large number of proposals of AI algorithms for different problems in the RAN, little attention has been paid to their practical implementation. This paper intends to fill this gap by discussing the practical aspects on the software development and implementation of AI algorithms for the RAN. This is done based on a specific example that uses deep reinforcement learning for the capacity sharing problem in RAN slicing. The paper presents an implementation of this solution in the context of the O-RAN architecture, detailing the operation of the involved interfaces and the containerization of the solution.**

I. Introducción

En sus cerca de cuarenta años de historia, los sistemas de comunicaciones móviles se han erigido en una parte fundamental de nuestra sociedad en tanto que permiten el acceso a servicios de comunicaciones de muy diversa índole, que progresivamente se han convertido en indispensables para la vida cotidiana. Por otra parte, los más recientes sistemas de quinta generación (5G) han ampliado la gama de servicios ofrecidos hacia nuevos sectores de aplicación, comúnmente denominados sectores verticales, y que engloban a modo de ejemplo las comunicaciones en entornos industriales, las ciudades inteligentes, las comunicaciones entre vehículos, etc.

La evolución de las redes de comunicaciones móviles ha venido condicionada, por un lado, por los cada vez más estrictos requerimientos de los servicios en términos de velocidad de transmisión, retardo o densidad de terminales conectados, por citar algunos ejemplos y, por otro lado, por la flexibilidad necesaria para proporcionar dichos servicios de forma eficiente. Con todo ello, las redes de comunicaciones móviles han visto incrementada enormemente su complejidad, de manera que la incorporación de mecanismos de automatización en la gestión y el control de las redes móviles se ha convertido en una necesidad imperiosa para el operador de la red. Esta misma complejidad es la que motiva la introducción de capacidades de aprendizaje máquina e Inteligencia Artificial (IA) en la red 5G en general y en la red de acceso radio (RAN: Radio Access Network) en particular. Un claro ejemplo de esta tendencia es la iniciativa Open Radio Access Network (O-RAN) (https://www.o-ran.org/), promovida recientemente por la O-RAN Alliance para la realización de una RAN virtualizada y cuya arquitectura incorpora dos controladores inteligentes de la RAN pensados para el soporte de algoritmos de IA.

Las aplicaciones de IA para la RAN abarcan múltiples niveles, que incluyen la capa física (p.ej. detección de señales MIMO), la asignación de recursos en la capa de control de acceso al medio MAC (p.ej, scheduling), la gestión de movilidad (p.ej. handover) o los sistemas de gestión y orquestación (p.ej. slicing). Así, si bien es de esperar que las soluciones de IA para las capas física y MAC, que operan en escalas temporales muy pequeñas, del orden del milisegundo, sean implementadas directamente por los fabricantes de los equipos de la RAN, otras funcionalidades que trabajan a escalas temporales superiores pueden ser incorporadas en entidades específicas, tales como los controladores inteligentes de la arquitectura O-RAN. Además, dichos controladores soportan la ejecución de aplicaciones (denominadas xApps y rApps) que pueden ser desarrolladas por terceros y que, en consecuencia, constituyen nuevas oportunidades de mercado en el ámbito de la IA para la RAN.

En este contexto, hay una extensa literatura en la que se presentan innumerables propuestas de algoritmos de IA para diferentes problemáticas de 5G [1][2]. Sin embargo, los aspectos de la implementación práctica de estos algoritmos de IA han recibido en general poca atención. Este artículo pretende contribuir a llenar este vacío en el estado del arte, discutiendo los aspectos prácticos del desarrollo del software y de la implementación de algoritmos de IA para la RAN. Esto se hará de la mano de un ejemplo específico, que trata la problemática de la compartición de capacidad para soportar "RAN slicing".

El resto del artículo está organizado como sigue. La sección II presenta los elementos principales de la arquitectura O-RAN en relación con la solución de compartición de capacidad considerada. La sección III discute los aspectos a tener en cuenta para el desarrollo software de los algoritmos IA. A su vez, la sección IV presenta la implementación de la interfaz que soporta la solución, y la sección V discute la implementación de la misma mediante contenedores. Finalmente, la sección VI presenta el repositorio software con la solución y la sección VII resume las conclusiones del artículo.

II. Marco de la Arquitectura O-RAN para la Solución de Compartición de Capacidad

La iniciativa O-RAN Alliance se inició en 2018 con objeto de estandarizar una arquitectura RAN que complementa la del 3GPP con un conjunto de interfaces abiertas para la realización de una RAN virtualizada y desagregada que incorpora funcionalidades de IA [3]. La Fig. 1 muestra la arquitectura lógica de O-RAN incluyendo sus componentes principales, resaltándose en color azul aquellos que son de especial interés en este artículo.

La arquitectura está compuesta por funciones O-RAN desagregadas. Éstas incluyen la unidad de radio O-RAN (O-RU), la unidad distribuida O-RAN (O-DU) y la unidad centralizada O-RAN, que a su vez se desgrana en la unidad del plano de control (O-CU-CP) y la del plano de usuario (O-CU-UP). Estas unidades llevan a cabo la funcionalidad de un

gNodoB (gNB) en 3GPP para el soporte de la tecnología 5G New Radio (NR) mediante una o más células (o celdas), cada una de las cuales proporciona cobertura en una zona geográfica a una cierta frecuencia. De entre las funciones del gNB, las unidades O-CU-CP y O-CU-UP se encargan de los niveles superiores de la interfaz radio, esto es, Radio Resource Control (RRC), Service Data Adaptation Protocol (SDAP) y Packet Data Convergence Protocol (PDCP). Estas funciones incluyen diferentes aspectos como p.ej. el control de la movilidad, la gestión de los reportes de medidas de los terminales o la compresión de cabeceras IP. Del mismo modo, la unidad O-DU se encarga de los niveles Radio Link Control (RLC), Medium Access Control (MAC) y de parte del nivel físico, incluyendo funciones tales como la gestión de las retransmisiones, la asignación dinámica de recursos (scheduling) o la codificación de canal, mientras que la O-RU lleva a cabo el resto de funciones del nivel físico y de RF, como p.ej. los procesos de IFFT/FFT o la amplificación de potencia. A su vez, la unidad O-RAN eNodoB (O-eNB) proporciona el soporte de la tecnología LTE. La función Service Management and Orchestration (SMO) se encarga de la gestión de estas unidades mediante la interfaz O1 y de la gestión del denominado O-cloud, que incluye la infraestructura de computación para soportar las diferentes funciones virtualizadas, mediante la interfaz O2.

Para efectuar la optimización de la RAN, la arquitectura O-RAN incluye dos controladores inteligentes (RIC: RAN Intelligent Controller) que permiten soportar funcionalidades de IA. El near-real-time RIC (near-RT RIC) permite efectuar el control de funciones con periodicidad de entre 10 ms y 1s mediante la interacción con las unidades O-CU-CP/UP y O-DU a través de la interfaz E2. A su vez, la función non-real-time RIC (non-RT RIC), incluida en el SMO, permite el control de funciones con escalas temporales superiores a 1s. El non-RT RIC también proporciona servicios de gestión de políticas y de gestión de modelos de Machine Learning (ML) en el near-RT RIC mediante la interfaz A1.

Tanto el near-RT RIC como el non-RT RIC soportan la ejecución de aplicaciones de terceros, denominadas respectivamente xApps y rApps. En concreto, las rApps del non-RT RIC pueden utilizarse para implementar funcionalidades como, por ejemplo, la gestión de frecuencias, la gestión de interferencias, etc. En general, la operativa de una rApp se sustenta en recoger diferentes medidas de prestaciones de las diferentes funciones O-RAN (p.ej. O-CU-CP/UP, O-DU, etc.) que reflejen el estado de las mismas y en procesarlas de acuerdo con un algorítmica, p.ej. de IA, que determine el valor de uno o varios parámetros de configuración de dichas

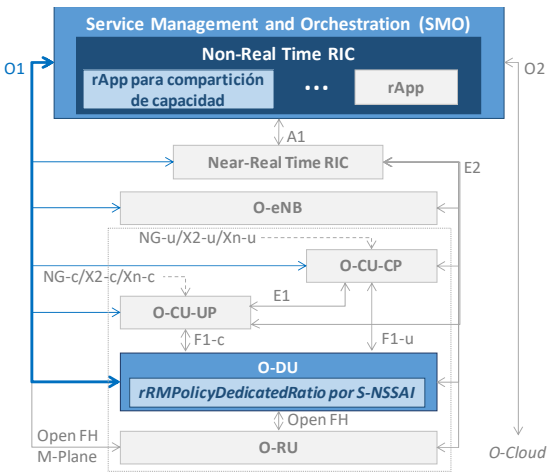

Fig. 1 Arquitectura O-RAN con la solución de compartición de capacidad.

funciones. Tanto la recogida de medidas como la configuración de parámetros se lleva a cabo mediante la interfaz O1.

La problemática de compartición de capacidad para RAN slicing considerada en el ejemplo de este artículo pretende ajustar dinámicamente la cantidad de recursos que se asignan a un RAN slice en cada una de las células que conforman la RAN que da servicio a una cierta área geográfica. Esto se lleva a cabo teniendo en cuenta los acuerdos a nivel de servicio (SLA: Service Level Agreement) para el RAN slice, en términos de una cierta capacidad (en b/s) agregada a proporcionar en la zona y una máxima capacidad por célula.

Tal como se ilustra en la Fig. 1, la funcionalidad de compartición de capacidad para RAN slicing puede implementarse como una rApp que ajusta la cantidad de recursos asignada a cada slice en cada célula en la unidad O-DU. El algoritmo, cuyos detalles pueden encontrarse en [4], está basado en una estrategia Multi-Agente con Reinforcement Learning (MARL) que utiliza la técnica Deep Q-Network (DQN).

La Fig. 2 ilustra las entradas y salidas del algoritmo, denominado DQN-MARL. En cuanto a las entradas, se utilizan métricas de prestaciones que recopila la O-DU y que están estandarizadas en [5], por lo que pueden ser obtenidas por el SMO a través de la interfaz O1. En concreto, se incluye, para cada célula, las métricas "Mean DL PRB used for data traffic", que mide el promedio de Physical Resource Blocks (PRBs) utilizados por slice, identificado por su Single Network Slice Assistance Information (S-NSSAI), en un período de medida, y "DL total available PRB", que proporciona el número medio de PRBs disponible. Estas dos métricas se emplean para determinar la fracción de ocupación de PRBs del slice, que se utiliza como una de las variables del estado de la técnica DQN-MARL (ver detalles en [4]). Por otra parte, el algoritmo también emplea como entrada la métrica "DL PDCP PDU Data Volume", que proporciona el volumen de datos por slice transferido en el enlace descendente entre el O-CU y el O-DU. Esta métrica se utiliza para determinar el throughput por slice, utilizado por el algoritmo al hacer la asignación de capacidad.

Como resultado de la ejecución del algoritmo DQN-MARL se obtiene la fracción de los PRBs disponibles que se asignan a cada slice en cada célula. La configuración de esta fracción de recursos por slice (S-NSSAI) se efectúa mediante el atributo de la función O-DU denominado *rRMPolicyDedicatedRatio* que se encuentra definido en el modelo de recursos de red del 3GPP [6].

III. Desarrollo de Software de Algoritmos IA

El desarrollo y la posterior implantación práctica de un algoritmo de IA implica diferentes fases. En primer lugar, es preciso efectuar el diseño del algoritmo en base a la problemática a la que debe hacer frente. Este diseño implica seleccionar la técnica de IA más apropiada, así como sus entradas y salidas.

Una vez efectuado el diseño, es preciso efectuar su desarrollo en software. Para ello, existen diferentes entornos con librerías que incorporan funciones específicas de IA, tales como el Deep Learning Toolbox de Matlab (https://es.mathworks.com/products/deep-learning.html) o la plataforma de código abierto Tensorflow

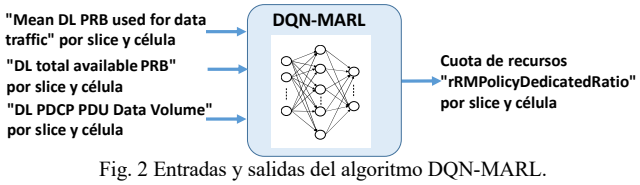

Fig. 2 Entradas y salidas del algoritmo DQN-MARL.

(https://www.tensorflow.org/). Para el ejemplo de este artículo, la solución DQN-MARL se ha implementado en Python mediante el entorno Tensorflow, haciendo uso de la librería TF-Agents [7], específicamente diseñada para algoritmos de reinforcement learning (RL) y que incluye la funcionalidad de DQN.

Otro aspecto importante del desarrollo es el entrenamiento de la solución, que consiste en ajustar los diferentes parámetros del algoritmo (p.ej. en el caso de una solución basada en DQN los valores de los pesos que definen las interconexiones dentro de la red neuronal) para que sea capaz de proporcionar las salidas adecuadas a las diferentes situaciones que puede encontrarse. En el caso de DQN-MARL, este entrenamiento se basa en múltiples interacciones con el entorno de operación, que consisten en configurar dicho entorno de acuerdo con el resultado que proporciona la red neuronal, esto es el valor de *rRMPolicyDedicatedRatio*, ver el impacto en términos de prestaciones de dicha configuración, el cual se traduce en una función de recompensa, y a partir de aquí actualizar los pesos de la red neuronal adecuadamente. En tanto que no resulta efectivo realizar dichas interacciones sobre una red real, ya que el efecto prueba y error que conllevan podría dar lugar a grandes degradaciones, resulta conveniente entrenar el sistema sobre un entorno de simulación que emule el comportamiento de la red real.

Finalmente, una vez la solución está entrenada y se ha validado su comportamiento sobre el entorno de simulación, se estaría en disposición de implantarla sobre el sistema real. En el ejemplo que nos ocupa, la solución DQN-MARL entrenada se desplegaría en la correspondiente rApp del non-RT RIC.

IV. Implementación de la Interfaz O1

La rApp de compartición de capacidad controla un determinado número de células asociadas con sus O-DU correspondientes. Por este motivo, es preciso implementar la interfaz O1 con cada una de estas O-DUs, tal y como se muestra en la Fig. 3. Sobre esta interfaz se despliegan dos servicios de gestión, o management services (MnS) [8]: el servicio de aprovisionamiento, "*Provisioning MnS*", que permite configurar el atributo *rRMPolicyDedicatedRatio* por S-NSSAI en el O-DU, y el servicio de aseguramiento de las prestaciones, "*Performance Assurance MnS*", que permite recopilar las métricas de prestaciones generadas por el O-DU que se utilizan como entrada en la solución DQN-MARL (ver [4]). Para ambos servicios de gestión la *rApp* actúa como consumidor o "MnS Consumer", mientras que la O-DU actúa como productor o "MnS Producer".

El "*Provisioning MnS*" se sustenta en el protocolo NETCONF, que define un mecanismo simple para gestionar dispositivos de red como p.ej. el O-DU. Mediante NETCONF es posible obtener parámetros de configuración del dispositivo

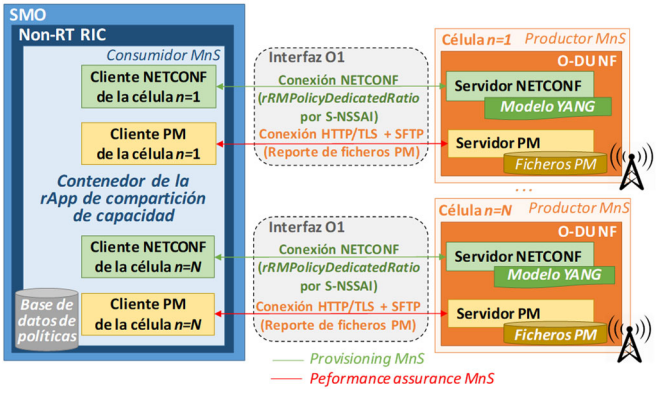

Fig. 3 Implementación de la interfaz O1 en la solución DQN-MARL.

gestionado, así como modificarlos o cargar nuevas configuraciones [9]. NETCONF utiliza una arquitectura de comunicación cuyo núcleo es la capa Remote Procedure Call (RPC) transportada sobre protocolos como p.ej. SSH, TLS, SOAP o BEEP, siendo SSH obligatorio. NETCONF opera siguiendo un esquema cliente-servidor en el que el "MnS Provider" actúa como servidor y el "MnS Consumer" como cliente. Así pues, como se observa en la Fig. 3, la rApp de compartición de capacidad incluye un cliente NETCONF para cada O-DU, encargado de establecer la conexión con el correspondiente servidor NETCONF en el O-DU y así configurar el atributo *rRMPolicyDedicatedRatio*.

El protocolo NETCONF define una serie de operaciones básicas que incluyen la operación de edición de configuración o "edit-config" , que permite al "MnS Consumer" modificar la configuración de atributos en el "MnS Producer" mediante un fichero codificado en XML con los valores para cada uno de los atributos siguiendo el procedimiento mostrado en la Fig. 4. Como se observa, tras establecer la sesión NETCONF, el "MnS Consumer" envía una petición "RPC request" al "MnS Provider" para editar la configuración de los atributos de la instancia del objeto gestionado, o "Managed Object Instance (MOI)". Los atributos del MOI están definidos mediante el lenguaje de modelado de datos YANG, que permite expresar la estructura y semántica de la información de configuración en un formato independiente del fabricante y de una forma compacta y legible. En el ejemplo considerado, el fichero de configuración XML configura los atributos de los módulos 3GPP YANG estandarizados en [6] que incluyen el atributo *rRMPolicyDedicatedRatio* para cada S-NSSAI. Tras recibir el "RPC request", el "MnS Provider" responde confirmando la configuración correcta del MOI o bien notificando de un error en caso de que la configuración no pudiera completarse. Finalmente, se termina la conexión NETCONF.

El "*Performance Assurance MnS*" se utiliza para reportar medidas de prestaciones o "Performance Measurements" (PM) del O-DU ("MnS Provider") a la *rApp* de compartición de capacidad ("MnS consumer"). Esto puede realizarse bien a través del envío de un fichero de datos de PM, lo que permite obtener las medidas en períodos del orden de minutos, o bien mediante la transmisión continua o "streaming" de PM, lo que permite una obtención de medidas en tiempo real. Para la solución de compartición de capacidad la implementación se ha realizado mediante un fichero de datos de PM, en tanto que la operación de la solución trabaja con periodicidades del orden de minutos. En base a esto, los ficheros de datos de PM se obtienen siguiendo el procedimiento mostrado en la Fig. 5. Cada vez que un nuevo fichero de datos de PM está disponible en el "MnS Provider", éste envía una notificación asíncrona *notifyFileReadyNotification* al "MnS Consumer" utilizando el protocolo HTTP/TL e indicando el nombre y la ubicación del nuevo fichero de datos de PM. A partir de la ubicación

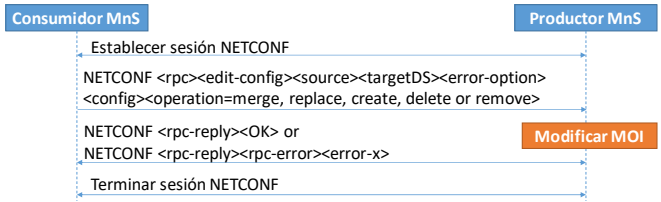

Fig. 4 Procedimiento simplificado de modificación de atributos.

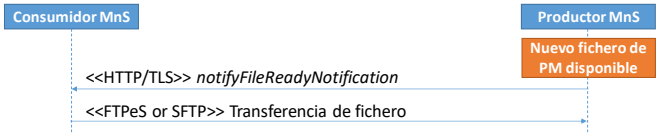

Fig. 5 Procedimiento de reporte de ficheros de datos de PM.

indicada, el "MnS Consumer" puede obtener el fichero mediante una transferencia SFTP o STPeS.

En el caso de la rApp de compartición de capacidad, para soportar el "*Performance Assurance MnS*" y establecer las conexiones HTTP/TL y SFTP necesarias, la rApp contiene un cliente PM para cada O-DU (ver Fig. 3). Este cliente obtiene los ficheros de datos PM del servidor PM en el O-DU. El fichero de datos de PM es un fichero XML definido de acuerdo con los formatos estandarizados por el 3GPP en [10] y las definiciones de las métricas de PM de [5], que incluyen, entre otras, las métricas de entrada al algoritmo DQN-MARL explicadas en la sección II (ver Fig. 2).

V. Contenerización de la Solución

La rApp de compartición de capacidad está implementada mediante contenerización, de modo que pueda ser ejecutada como una unidad software aislada denominada contenedor o "container" que empaqueta todos los componentes necesarios para la rApp, esto es, el código, librerías, ficheros binarios, ficheros de configuración y dependencias [11]. De este modo el contenedor puede ser encapsulado y desplegado fácilmente en el non-RT RIC del SMO, donde las diferentes rApp pueden ejecutarse de forma aislada al resto de contenedores, compartiendo el mismo sistema operativo. La gestión de los diferentes contenedores se lleva a cabo mediante el motor de contenedores *Docker*.

El código software del contenedor de la rApp de compartición de capacidad incluye la importación y aplicación las redes neuronales previamente entrenadas (denominadas "políticas") asociadas a cada uno de los slices de acuerdo con la solución DQN-MARL, así como los clientes NETCONF y PM para la interfaz O1 explicados en la sección III. El *Algoritmo* 1 resume la operación del código desarrollado. Tras establecer las sesiones NETCONF y HTTP/TLS con los O-DU involucrados, se carga la política de cada una de las slices a partir de las políticas disponibles en una base de datos (líneas 1-3 del algoritmo). A partir de este momento, de forma periódica se reciben los ficheros de datos PM desde el servidor de cada O-DU y a partir de las métricas disponibles se calcula el estado de cada slice (líneas 5-7). Entonces se aplica la política de decisión del slice para ese estado, esto es, se ejecuta la red neuronal tomando como entrada los valores del estado y a partir de las salidas se determina el valor de *rRMPolicyDedicatedRatio* para cada slice y célula. Este valor se configura en los diferentes O-DUs mediante el procedimiento NETCONF de modificación de atributos (líneas 8-9). Las operaciones de las líneas 5 a 9 del algoritmo se repiten cada Δt minutos. El código software del *Algoritmo* 1 se ha desarrollado en Python usando la librería TF-agents. Este código se ha incluido en el contenedor conjuntamente con las dependencias requeridas para su operación, el acceso a la base de datos de políticas, la versión de Python y las librerías necesarias.

***Algoritmo* 1 –rApp de compartición de capacidad**
1 Establecer la sesión NETCONF para cada O-DU.
2 Establecer la sesión HTTP/TLS para cada O-DU.
3 Cargar la política de cada slice.
4 Bucle periódico:
5 Recibir *notifyFileReady* vía HTTP/TLS de cada O-DU.
6 Obtener el fichero de medidas PM vía SFTP para cada O-DU.
7 Calcular el estado para cada S-NSSAI mediante las medidas PM.
8 Obtener el valor del *rRMPolicyDedicatedRatio* para cada S-NSSAI y célula aplicando la política.
9 Enviar el valor de *rRMPolicyDedicatedRatio* para cada S-NSSAI y célula mediante una petición “edit-config” de NETCONF.

VI. Resultados Software

Como resultado del trabajo de este artículo se ha creado un repositorio de software abierto que incluye la solución software del algoritmo DQN-MARL, así como de los elementos necesarios para la implementación de la interfaz O1 y la contenerización de la solución. El repositorio software está disponible en la plataforma GitHub accesible mediante el sitio web del proyecto PORTRAIT (https://portrait.upc.edu/). Además del código fuente de los diferentes elementos solución y del entorno de testeo, el repositorio proporciona una serie de documentación a modo de tutorial que guían al usuario por la instalación y ejecución de los diferentes elementos de la solución.

VII. Conclusiones

Este artículo ha discutido diferentes aspectos del desarrollo software e implementación de soluciones de inteligencia artificial para la gestión de redes radio 5G. La discusión ha tomado como ejemplo una solución específica para la compartición de capacidad en entornos con “RAN slicing” compatible con O-RAN, para la cual se ha descrito el desarrollo software necesario, así como la implementación de las interfaces requeridas para la interacción de la solución con la red 5G y su contenerización para ser ejecutada en una plataforma compatible con O-RAN. Con el fin de proporcionar un ejemplo práctico ilustrativo para la comunidad científica de este campo, la discusión del artículo se complementa con un repositorio software de código abierto que incluye los diferentes elementos de la solución y su correspondiente documentación.

Agradecimientos

Este trabajo es parte del proyecto PORTRAIT (ref. PDC2021-120797-I00) financiado por MCIN/AEI/10.13039/501100011033 y por European Union Next Generation EU/PRTR.

Referencias

[1] Y. Sun, et al. "Application of Machine Learning in Wireless Networks: Key Techniques and Open Issues," *IEEE Comms. Surveys & Tutorials*, vol. 21, no. 4, pp. 3072-3108, 4th Quarter 2019.

[2] J. Wang, et.al, “Thirty Years of Machine Learning: The Road to Pareto-Optimal Wireless Networks” *IEEE Comms. Surveys & Tutorials,* Vol. 22, No. 3, 3rd Quarter, 2020.

[3] M. Polese, et. al. “Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges,” arXiv:2202.01032 [cs.NI], 2022. [Online].

[4] I. Vilà, et.al, “A Multi-Agent Reinforcement Learning Approach for Capacity Sharing in Multi-tenant Scenarios,” in *IEEE Trans. Veh. Tech.*, vol. 70 no. 9, July 2021.

[5] 3GPP TS 28.552 v16.2.0, “Management and orchestration; 5G Performance measurements (Release 16),” June 2019.

[6] 3GPP TS 28.541 v16.0.0, “Management and orchestration; 5G Network Resource Model (NRM) (Release 16),” March 2019.

[7] S. Guadarrama, et. al. "TF-Agents: A library for Reinforcement learning in TensorFlow.", 2018. Available at: https://github.com/tensorflow/agents.

[8] O-RAN.WG10.O1-Interface.0-v05.00, “O-RAN Operations and Maintenance Interface Specification,” O-RAN Alliance, Working Group 10, Technical Specification, August 2020.

[9] J. Schönwälder, et. al. "Network configuration management using NETCONF and YANG," in *IEEE Communications Magazine*, vol. 48, no. 9, pp. 166-173, Sept. 2010.

[10] 3GPP TS 28.550 v16.7.0, “Management and orchestration; Performance assurance (Release 16),” Dec. 2020.

[11] A. Gopalasingham, et. al. "Virtualization of radio access network by Virtual Machine and Docker: Practice and performance analysis," 2017 *IFIP/IEEE Symp. on Int. Net. and Serv. Manag. (IM)*, 2017.