Plan de Recuperación, Transformación y Resiliencia

Financiado por la Unión Europea NextGenerationEU

uc3m | Universidad Carlos III de Madrid

SORUS

Validación y optimización conjunta de RIS y vRANs

SORUS-RAN A2.3-E1

PRIMERA VERSIÓN DEL PROTOTIPO PARA VALIDAR LA ORQUESTACIÓN VRAN

Revisión	Autor	Fecha de entrega	Cambios
Versión 01	Jose Ayala Romero, Andrés García Saavedra	10/09/2023	Versión Inicial
Versión 02	Luis Roda Sánchez, Jorge San Martin Gomez	21/09/2023	revisión y actualización de tablas y resumen ejecutivo y conclusiones.

Exención de responsabilidad:

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

CONTENIDOS

LISTA DE ABREVIATURAS Y ACRÓNIMOS

Tabla 1. Lista de abreviaturas y acrónimos

Abreviatura	Explicación/Definición
BBU	BaseBand Unit
BP-vRAN	Bayesian optimization for Power consumption in vRANs
BS	Base Station
DDPG	Deep Deterministic Policy Gradient
DL	Downlink
DU	Distributed Unit
eNB	eNodeB
gNB	gNodeB
GP	Gaussian Process
GPU	Graphics Processing Unit
HARQ	Hybrid Automatic Repeat Request
IA	Inteligencia Artificial
IoT	Internet of Things
LA	Learning Agent
LDPC	Low-Density Parity Check
LoS	Line of Sight
MAC	Media Access Control
mAP	mean Average Precision
MCS	Modulation Coding Scheme
MIMO	Multiple-Input Multiple-Output
MRT	Maximal Ratio Transmission
MSE	Mean Squared Error
NLoS	Non-Line of Sight
NN	Neural Network
NR	New Radio

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

Abreviatura	Explicación/Definición
OFDMA	Orthogonal Frequency-Division Multiple Access
PBCH	Physical Broadcast Channel
PCFICH	Physical Control Format Indicator Channel
PDCCH	Physical DL Control Channel
PoE	Power-over-Ethernet
PRACH	Physical Random Access Channel
QoS	Quality of Service
RAN	Radio Access Network
RF	Radio Frequency
RIS	Reconfigurable Intelligent Surface
RKHS	Reproducing Kernel Hilbert Space
RL	Reinforcement Learning
RLC	Radio Link Control
RU	Radio Unit
SBP-vRAN	Safe Bayesian optimization for Power consumption in vRANs
SC-FDMA	Single-Carrier Frequency-Division Multiple Access
SDR	SemiDefinite Relaxation
SF	SubFrame
SINR	Signal to Interference & Noise Ratio
SMSE	Sum of Mean Squared Errors
SNR	Signal-to-Noise Ratio
TB	Transport Block
TTI	Transmission Time Interval
UCB	Upper Confidence Bound
UCI	Uplink Control Information
UE	User Equipment

Abreviatura	Explicación/Definición
UL	Uplink
UPA	Uniform Planar Array
vBS	virtualized Base Station
vDU	virtualized DU
VNF	Virtual Network Function
vRAN	virtualized Radio Access Network

LISTA DE FIGURAS

LISTA DE TABLAS

1 RESUMEN EJECUTIVO

Este documento conforma el entregable SORUS-RAN-A2.3, en el que se plantean diferentes algoritmos para dar solución a los tres casos de uso planteados en el entregable SORUS-RAN-A1.2. La finalidad de los mismos es la de explorar distintos escenarios para optimizar el consumo energético de las estaciones base virtualizadas, manteniendo los requerimientos de calidad de servicio. El objetivo es diseñar un sistema de políticas de energía capaz de adaptarse a las necesidades del usuario y a las condiciones de la red maximizando el rendimiento del sistema. Se han hecho experimentos tanto en escenarios estáticos como dinámicos para calcular el rendimiento de la propuesta en presencia de rápida variabilidad y cambios de repentinos de restricción.

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

2 INTRODUCCIÓN

En este entregable partimos de los tres casos de uso propuestos en SORUS-RAN-A1.2 y proponemos algoritmos que los abordan. Como se desarrolla en SORUS-RAN-A1.2, a diferencia de las estaciones base (BS) tradicionales, las BS virtualizadas (vBS) tienen un perfil de rendimiento y consumo energético complejo, poliparamétrico y dependiente de la plataforma hardware, lo que hace que las políticas de control tradicionales resulten ineficaces para su gestión.

Con el fin de superar este obstáculo, proponemos y evaluamos un novedoso framework de aprendizaje automático que aprende *on-the-fly* los perfiles operativos de los vBS y selecciona su configuración óptima en función de las necesidades de la red y de la disponibilidad o restricciones de energía. En particular, proponemos tres algoritmos. Para el caso de uso 1, el algoritmo permite a los operadores equilibrar el rendimiento y los gastos de energía. En el segundo caso de uso, hay un tope máximo de energía a usar, lo cual es crucial para las vBS que se ejecutan en plataformas con restricciones de energía, por ejemplo, las células Power-over-Ethernet (PoE). Finalmente, en el caso de uso 3, proponemos un algoritmo que optimiza de forma conjunta la vBS y un servicio de inteligencia artificial al borde de la red, con el objetivo de minimizar energía mientras satisface los requerimientos de calidad de servicio (QoS) en el borde de la red.

Nuestros algoritmos se basan en la teoría de optimización bayesiana [1] y Procesos Gaussianos (GPs) [2]. Estas herramientas son apropiadas para nuestros problemas debido a que, como mostramos en este trabajo, son especialmente eficientes en datos, lo cual es un requisito importante en nuestro caso dada la naturaleza altamente dimensional de nuestro espacio de contexto-acción.

Las GP modelan el comportamiento del vBS en términos de rendimiento y consumo de energía, utilizando mediciones que se recogen en tiempo de ejecución. El problema se formula como un contextual badit (ver SORUS-RAN-A1.2) con el objetivo de explorar el espacio de configuraciones de la vBS y explotar las mejores para cada contexto. Para esto último, utilizamos los valores medios de carga de tráfico Uplink (UL)/Downlink (DL) y ratio señal-ruido (SNR), que medimos a lo largo de ciertas ventanas temporales, ya que vienen determinados por la especificación 3GPP O-RAN pertinente [6]. El resultado es un marco algorítmico no paramétrico que hace suposiciones mínimas sobre el sistema, se adapta a las necesidades del usuario y a las condiciones de la red, y maximiza el rendimiento del sistema. Así, este framework de aprendizaje supera a otros enfoques que requieren el conocimiento de las funciones vBS [3] o datos previos para aproximarlas [7], y técnicas adaptativas que no ofrecen garantías de rendimiento o se basan en supuestos estrictos de modelado del sistema [4][5].

Por último, realizamos una evaluación exhaustiva en una plataforma experimental basada en srsRAN [8], y utilizando varias herramientas para medir en tiempo real el consumo de energía de la vBS. Este es un paso importante en nuestro estudio, ya que nos permite evaluar la eficacia práctica de los algoritmos de aprendizaje propuestos. De hecho, comprobamos que ambas soluciones convergen a la configuración óptima de la vBS en diversos escenarios. Para ello, también proponemos y evaluamos varias mejoras prácticas que aceleran la convergencia de los algoritmos. Utilizando trazas de tráfico real, mostramos paso a paso cómo nuestro marco explora las configuraciones y cómo se abstiene de violar las restricciones en cada caso. También comparamos nuestra solución con una solución de aprendizaje por refuerzo (RL). En concreto, implementamos un algoritmo Deep Deterministic Policy Gradient (DDPG) utilizando una arquitectura de red neuronal (NN) actor-critic [9], y lo adaptamos a nuestro problema de contextual bandit.

Descubrimos que nuestro marco es más eficiente en datos que otros enfoques basados en RL, que requieren órdenes de magnitud de más mediciones (por lo tanto, también más tiempo) para entrenar las NN.

3 ALGORITMOS

Partiendo de los casos de uso presentados en el entregable SORUS-RAN-A1.2, en este entregable se proponen algoritmos para cada uno de esto casos de uso que a continuación se resumen brevemente. Para más detalle, consulte SORUS-RAN-A1.2.

- **Caso de uso 1:** equilibrio entre rendimiento y coste. Consideramos una vBS compuesta por una unidad de banda base (BBU) que puede corresponder a un eNB 4G o a un gNB 5G alojado en una plataforma de computación y conectado a una unidad de radio (RU). Este tipo de BS es relevante para células pequeñas de bajo coste, células PoE y otras plataformas similares que son cada vez más comunes en las redes 5G y posteriores.
- **Caso de uso 2**: limitación en la máxima potencia que la vBS puede usar. En este caso de uso consideramos que el vBS se alimenta de una fuente de energía limitada en potencia como puede ser una batería o una fuente PoE.
- **Caso de uso 3**: optimización conjunta de vBS y servicios Edge de Inteligencia Artificial (IA). Consideramos un servidor al borde de la red (*network edge*) que dispone de una GPU que proporciona un servicio de IA a través de una red de acceso por radio. El servicio de IA puede ser, por ejemplo, un servicio de reconocimiento de objetos que puede utilizarse para la vigilancia de la seguridad o la detección de fallos en cadenas industriales. Suponemos que se crea un slice dedicado a este servicio que incluye la vBS y el servidor Edge.

3.1. Caso de uso 1

Muchos algoritmos para resolver problemas de contextual bandits asumen que hay un vector de características asociado a cada acción, y que la función objetivo es lineal en ese vector [10],[11]. Este supuesto no se cumple aquí por las siguientes razones: en primer lugar, la función objetivo no es lineal (ver definición en SORUS-RAN-A1.2); en segundo lugar, los valores de la función asociados con diferentes acciones (es decir, las políticas de control vBS) están correlacionados. Intuitivamente, podemos pensar que un pequeño cambio en algún parámetro (por ejemplo, el airtime) inducirá un pequeño cambio en la potencia consumida por el vBS. Esto significa que podemos obtener información sobre pares contexto-control no observados mediante la observación de acciones cercanas, reduciendo así el tiempo de exploración.

Basándonos en estas observaciones, proponemos un método de optimización bayesiano en el que modelamos la función objetivo como una muestra de un GP sobre el espacio conjunto contexto-control. Este estimador no paramétrico captura las no linealidades y correlaciones antes mencionadas, y proporciona incertidumbre predictiva sobre la estimación de la función. Por lo tanto, nos permite abordar eficazmente la disyuntiva entre exploración y explotación.

3.1.1. Estimador de funciones

Utilizamos una GP como estimador de función, que es una colección de variables aleatorias que siguen distribuciones gaussianas conjuntas [2]. Definimos $z \in \mathcal{Z} = \Omega \times \mathcal{X}$ como un par contexto-control. Modelamos la función objetivo como una muestra de un $GP\big(\mu(z), k(z,z')\big)$, donde $\mu(z)$ es su función media y $k(z,z')$ es su función de covarianza o kernel. Sin pérdida de generalidad, suponemos $\mu = 0$ y varianza acotada $k(z,z) < 1$, a la que nos referimos como la distribución *a priori*, no condicionada a los datos.

Dada esta distribución a priori y un conjunto de observaciones, la media y la covarianza de la distribución a posteriori pueden calcularse utilizando fórmulas cerradas. Sea $y_T = [u_1, \dots, u_T]$ un vector de muestras ruidosas (suponiendo *i.i.d.* ruido gaussiano $\sim N(0, \zeta^2)$) en los puntos $Z_T = [z_1, \dots, z_T]$. Entonces, la distribución posterior de la función objetivo sigue una distribución GP con media $\mu_T(z)$ y covarianza $k_T(z, z')$:

$$\mu_T(z) = k_T(z)^\top (K_T + \zeta^2 1_T)^{-1} y_T \quad (1)$$

$$k_T(z, z') = k(z, z') - k_T(z)^\top (K_T + \zeta^2 1_T)^{-1} k_T(z') \quad (2)$$

donde $k_T(z) = [k(z_1, z), \dots, k(z_T, z)]^\top$, $K_T(z)$ es la matriz kernel $[k(z, z')]_{z,z' \in Z_T}$, y 1_T es la matriz identidad de dimensión T. Estas ecuaciones nos permiten estimar la distribución de los valores no observados de z basándonos en la distribución a priori, el vector Z_T y las observaciones de la función y_T.

3.1.2. Función de kernel

La selección del kernel es crucial, ya que da forma a las distribuciones GP a priori y a posteriori codificando la correlación entre los valores de la función objetivo de cada par de puntos. En concreto, $k(z, z')$ indica la similitud entre la función objetivo evaluada en z y z'. En otras palabras, el kernel caracteriza la suavidad de la función [12].

Las propiedades de la función kernel deben seleccionarse cuidadosamente según la aplicación específica y la función subyacente que se aprenderá. Por lo tanto, utilizamos los datos experimentales analizados en SORUS-RAN-A2.1 para concluir que nuestro kernel debe satisfacer dos propiedades: *Estacionariedad y Anisotropicidad*. Por un lado, el kernel $k(z, z')$ es estacionario ya que sólo depende de la distancia de z a z', lo que significa que es invariante a traslaciones en $\mathcal{Z}$. Por otro lado, un kernel es anisótropo, ya que la *smoothness* de la función objetivo es diferente entre las diferentes dimensiones de $\mathcal{Z}$. Es decir, el núcleo no es invariante a las rotaciones en $\mathcal{Z}$. La smoothness de las diferentes dimensiones de la función objetivo se codifican en un vector de longitud-escala $\mathcal{L} = [l_1, \dots, l_N]$, donde N indica el número de dimensiones de $\mathcal{Z}$. Así, la distancia entre dos puntos basada en el vector longitud-escala puede escribirse como:

$$d(z, z') = \sqrt{(z - z')^\top L^{-2} (z - z')},$$

donde $L = \mathrm{diag}(\mathcal{L})$ es una matriz diagonal de los valores de escala de longitud.

Existen varias funciones kernel que satisfacen estas propiedades, como el kernel exponencial al cuadrado, uno de los más utilizados. Sin embargo, esta función kernel supone que la función subyacente es muy suave, es decir, infinitamente diferenciable. Este supuesto no se cumple en nuestro marco ya que la función $B(\cdot)$ definida SORUS-RAN-A1.2 no es infinitamente diferenciable. Además, recordemos que $B(\cdot)$ representa el coste monetario asociado a la potencia consumida y puede definirse en función de las necesidades del operador.

Por ello, relajamos este supuesto y seleccionamos la versión anisotrópica del kernel Matérn, que también satisface las propiedades antes comentadas [2]. Además, lo configuramos con el parámetro $\nu = \frac{3}{2}$, lo que implica que la función objetivo es al menos una vez diferenciable.

Obsérvese que se trata de una suposición suave, que produce un límite de regret poco estricto (véase el lema 1). De hecho, nuestra evaluación experimental en secciones posteriores muestra que nuestro enfoque funciona mucho mejor que nuestros límites teóricos en los escenarios que probamos. Sin embargo, si tuviéramos más información acerca de la estructura de la función a

aprender, podríamos fácilmente disminuir tal límite mediante la selección de valores más altos de ν o mediante el uso de un núcleo exponencial al cuadrado, lo que puede mejorar la tasa de aumento de la ganancia de información. En este trabajo, optamos por la opción más conservadora para cubrir escenarios más allá de los mostrados en nuestra evaluación experimental. La expresión del kernel seleccionado viene dada por:

$$k(z, z') = \left(1 + \sqrt{3}d(z, z')\right) \exp\left(-\sqrt{3}d(z, z')\right).$$

Para mejorar el rendimiento, podemos optimizar los hiperparámetros $\mathcal{L}$ y la varianza del ruido ζ^2, ecuaciones (1)-(2), antes de ejecutar el algoritmo, maximizando la estimación de la verosimilitud sobre los datos anteriores y mantener estos valores constantes en el tiempo. Un enfoque diferente, es decir, cuando los hiperparámetros se optimizan utilizando los datos adquiridos en tiempo de ejecución, no está garantizado que el intervalo de confianza de la GP cubrirá la verdadera función, y, por lo tanto, podría inducir el proceso de optimización a atascarse en óptimos locales [13], algo que también hemos observado en nuestros experimentos.

3.1.3. Función de adquisición

La función de adquisición selecciona un control x_t en cada periodo t basándose en la distribución posterior de la función objetivo sobre los pares contexto-control. Para ello, utilizamos el método Upper Confidence Bound (UCB) que sigue el principio de *optimismo ante la incertidumbre* y nos permite derivar garantías teóricas para el algoritmo. Formalmente:

$$x_t = argmax_{x \in \mathcal{X}}\ \mu_{t-1}(\omega_t, x) + \sqrt{\beta_t}\sigma_{t-1}(\omega_t, x), \quad (3)$$

donde ω_t es el contexto observado en el tiempo t, β_t es un parámetro de ponderación y $\sigma_t^2(z) = k_t(z, z)$. Formalizamos nuestro enfoque, al que nos referimos como BP-vRAN (Bayesian optimization for Power consumption in vRANs), en el Algoritmo 1.

Al principio de cada periodo de decisión t se observa un contexto ω_t (línea 4). En base al contexto observado ω_t y los vectores Z_{t-1} y y_{t-1}, la distribución posterior se calcula utilizando ecuaciones (1)-(2) (línea 5). Tenga en cuenta que cuando no tenemos datos (y_0, y Z_0 son set vacíos) la distribución posterior es igual a la distribución a priori. El control x_t se decide basándose en la GP posterior y la función de adquisición (línea 6). Al final de t, se observan el rendimiento y la potencia consumida (línea 7). A continuación, la recompensa y el coste monetario de la energía se calculan para posteriormente calcular el valor de la función objetivo de acuerdo a la definición dada en SORUS-RAN-A1.2 (línea 8). Finalmente, el nuevo par contexto-control z_t y el valor de la función objetivo $u_t(\omega_t, x_t)$ se incluyen en los vectores Z_t y y_t, respectivamente, para mejorar la distribución posterior de la siguiente iteración (líneas 9-10).

Algoritmo 1. BP-vRAN: Equilibrio entre rendimiento y costes
1: Inputs: Espacio de control $\mathcal{X}$, kernel k, β
2: Inicializar y_0 Z_0 como sets vacíos.
3: For $t = 1,2, \ldots$ do
4: Observar el contexto ω_t
5: Computar μ_{t-1} y $\sigma_{t-1}^2 = k_{t-1}(z_t, z_t)$, ecuaciones (1)-(2)

6:	$x_t = argmax_{x\in\mathcal{X}}\ \mu_{t-1}(\omega_t, x) + \sqrt{\beta_t}\sigma_{t-1}(\omega_t, x)$
7:	Medir $R_t^{dl}\left(\omega_t^{dl}, x_t^{dl}\right)$, $R_t^{ul}\left(\omega_t^{ul}, x_t^{ul}\right)$ and $P_t(\omega_t, x_t)$ al final del periodo de decisión t
8:	Computar $u_t(\omega_t, x_t)$
9:	Actualizar $Z_t \leftarrow Z_{t-1} \cup z_t \coloneqq [\omega_t, x_t]$
10:	Actualizar $y_t \leftarrow y_{t-1} \cup u_t(\omega_t, x_t)$
End for	

Obsérvese que una formulación alternativa de BP-vRAN con dos GP (para aproximar la recompensa y la potencia consumida por separado) en lugar de una permite optimizar mejor los hiperparámetros de los kernels. No obstante, la varianza posterior de la función objetivo puede ser arbitrariamente difícil de obtener, ya que el coste monetario de la potencia ($B(\cdot)$) es seleccionado por el operador en función de sus necesidades. Además, este enfoque duplica los requisitos computacionales y de memoria.

3.1.4. Resultados teóricos

La elección de un valor para β_t en la ecuación (3) es muy importante, ya que controla la compensación entre exploración y explotación. Valores altos de β_t llevan a la función de adquisición a seleccionar controles con mayor incertidumbre mientras que, por el contrario, los controles que ya se sabe que son de alto rendimiento (aunque no necesariamente de mayor rendimiento) se seleccionan cuando β_t toma valores más pequeños. Siguiendo [14], seleccionamos

$$\beta_t = 2B^2 + 300\gamma_t \ln^3(t/\epsilon),$$

donde $\epsilon \in (0,1)$, $B \geq |u|_k$ es un límite superior de la norma del espacio de Hilbert del núcleo reproductor (RKHS) de u (la función objetivo), y γ_t es la máxima ganancia de información mutua obtenida de u después de que se hayan recogido t observaciones.

Lema 1: el regret contextual R_T de BP-vRAN satisface:

$$\mathrm{P}\left(R_T \leq \sqrt{C_1 T \beta_T \gamma_T}\ \forall T \geq 1\right) \geq 1 - \epsilon,$$

en la etapa T, donde $C_1 = \frac{8}{\log(1+\zeta^{-2})}$ y $\gamma_t = \mathcal{O}\left(t^{44/45}\log(t)\right)$.

La prueba de *Lema 1* se basa en [15] y [14]. Para derivarla, en primer lugar, se definen los siguientes lemas.

Lema 2: sea $\epsilon \in (0,1)$, supongamos que el ruido en la observación es uniformemente acotado por ζ y $\beta_t = 2B^2 + 300\gamma_t \ln^3(t/\epsilon)$, entonces:

$$Pr\{\forall t,\ \forall z \in \mathcal{Z}, |\mu_{t-1}(z) - u(z)| \leq \sqrt{\beta_t}\sigma_{t-1}(z)\} \geq 1 - \epsilon.$$

Prueba. Dada en Teorema 6 en [15].

Lema 3: fijar $t \geq 1. Si\ |u(z) - \mu\_\{t-1\}(z)| \leq \sqrt{\{\beta\_t\}}\ \sigma\_\{t-1\}(z)$ para todo $z \in \mathcal{Z}$, el regret contextual r_t está entonces limitado por $2\sqrt{\{\beta\_t\}}\ (z\_t)$.

Prueba. La prueba sigue el Lema 4.1 en [14]. Sea $x_t^* \in \mathrm{argsup}_{x\in\mathcal{X}} u(\omega_t, x)$ el control óptimo en el periodo de decisión t. Entonces, considerando que x_t es el control seleccionado en el período de

decisión t, dada la función de adquisición en la ecuación (3): $\mu_{t-1}(\omega_t, x_t) + \sqrt{\beta_t}\sigma_{t-1}(\omega_t, x_t) \geq \mu_{t-1}(\omega_t, x_t^*) + \sqrt{\beta_t}\sigma_{t-1}(\omega_t, x_t^*) \geq u(\omega_t, x_t^*)$.

Entonces,

$r_t = u(\omega_t, x_t^*) - u(\omega_t, x_t) \leq \sqrt{\beta_t}\sigma_{t-1}(\omega_t, x_t) + \mu_{t-1}(\omega_t, x_t) - u(\omega_t, x_t) \leq 2\sqrt{\beta_t}\sigma_{t-1}(\omega_t, x_t)$.

Lema 4: la ganancia de información para los puntos seleccionados puede expresarse en términos de las varianzas predictivas. Si $u_T = \big(u(z_t)\big) \in R^T$ [15]:

$$I(y_T; u_T) = \frac{1}{2}\sum_{t=1}^{T} \log\big(1 + \zeta^{-2}\sigma_{t-1}^2(z_t)\big)$$

Prueba. Dada en Lema 5.3 en [15].

Prueba del Lema 1

La prueba sigue el Teorema 5 en [14]. Por Lema 2 y Lema 3, tenemos que $Pr\{r_t^2 \leq 4\beta_t\sigma_{t-1}^2(z_t)\forall t \geq 1\} \geq 1 - \epsilon$.

Dado que β_t es no decreciente, tenemos que

$$\begin{aligned} 4\beta_t\sigma_{t-1}^2(z_t) &\leq 4\beta_T\zeta^2\big(\zeta^{-2}\sigma_{t-1}^2(z_t)\big) \\ &\leq 4\beta_T\zeta^2 C_2 \log\big(1 + \zeta^{-2}\sigma_{t-1}^2(z_t)\big) \end{aligned} \quad (4)$$

con $C_2 = \zeta^{-2}/\log(1 + \zeta^{-2}) \geq 1$, ya que $s^2 \geq C_2 \log(1 + s^2)$ para $s \in [0, \zeta^{-2}]$, y $\zeta^{-2}\sigma_{t-1}^2(z_t) \leq \zeta^{-2}k(z_t, z_t) \leq \zeta^{-2}$.

Considerando $C_1 = 8\sigma^2 C_2$ and $R_T^2 \leq T\sum_{t=1}^{T} r_t^2$ (desigualdad de Cauchy-Schwarz), el resultado sigue del Lema 4.

Para la derivación de la cota de la ganancia de información γ_t, consideramos un kernel Matérn con $\nu = \frac{3}{2}$ y $N = 11$ dimensiones en $\mathcal{Z}$, que corresponden a un contexto y un espacio de control de 6 y 5 dimensiones, respectivamente. Para este entorno, particularizamos la expresión proporcionada en el Teorema 5 de [15] para obtener el límite $\gamma_t = \mathcal{O}\big(t^{44/45}\log(t)\big)$.

Nótese que el límite del regret obtenido en este análisis considera el peor de los casos, mientras que el rendimiento del algoritmo en la práctica suele estar lejos de estos límites como se muestra más adelante en la evaluación experimental. Vale la pena mencionar, sin embargo, que el límite proporcionado en el Lema 1 indica que BP-vRAN es un algoritmo de no-regret, es decir, $\lim_{T\to\infty} E\,[R_T]/T = 0$.

3.2. Caso de uso 2

La imposición de restricciones estrictas, como se propone en el segundo caso de uso, hace el problema más complicado. Trabajos anteriores, por ejemplo, en robótica y otras áreas [16],[18],[19],[20], han propuesto algoritmos bayesianos de optimización con restricciones seguras. Su idea principal radica en la definición: cada t definimos un subconjunto de controles *seguro* $S_t \subseteq \mathcal{X}$ que satisfagan las restricciones con certeza. A continuación, es necesario intercalar un proceso de exploración para ampliar el conjunto seguro, mientras se busca una acción segura con un alto rendimiento. Desgraciadamente, estos trabajos no tienen en cuenta la información contextual, que afecta claramente al conjunto seguro, es decir, $S_t(\omega_t) \subseteq \mathcal{X}$. Hasta donde sabemos, SafeOpt [16] es

el único trabajo que propone un algoritmo de aprendizaje contextual seguro. Sin embargo, aunque ese algoritmo ofrece garantías teóricas, su función de adquisición selecciona el control con mayor incertidumbre entre todos los candidatos que pueden ampliar el conjunto seguro y también los maximizadores potenciales. En nuestros experimentos comprobamos que este enfoque tiene una convergencia excesivamente lenta. Este problema práctico ha sido reportado en otros trabajos también, por ejemplo, [17]. Por lo tanto, mejoramos esta metodología mediante el empleo de la función de adquisición de CGP-UCB [14], pero restringido al conjunto seguro de controles.

Denotamos $y_T^f = [r_1, \dots, r_T]$ el vector de muestras de recompensa en T y $y_T^c = [P_1, \dots, P_T]$ las muestras de consumo de energía. Utilizamos una GP para la recompensa y otra para la potencia. Ambas GPs tienen la misma distribución a priori y kernel pero diferentes hiperparámetros. La distribución posterior puede calcularse utilizando las ecuaciones (1)-(2), y sustituyendo y_T por y_T^f o y_T^c, para cada GP. Denotamos la media posterior y la covarianza de la recompensa en T como $\mu_T^f(z)$ y $k_T^f(z, z')$, y $\mu_T^c(z)$ y $k_T^c(z, z')$ para la potencia, respectivamente. El conjunto seguro inicial $S_0 \subseteq \mathcal{X}$ es común para todos los contextos, e incluye configuraciones de bajo consumo de energía (vBS cerca del reposo). En el peor de los casos, S_0 puede ampliarse utilizando datos de la vBS.

En cada periodo, S_t se calcula a partir de la distribución posterior del consumo de energía proporcionada por la GP. Suponemos que el valor real del consumo de energía en el momento t está dentro del intervalo $[\mu_t^c(z) \pm \beta_t \sigma_t^c(z)]$, donde $\sigma_t^c(z) = k_t^c(z, z)$. Utilizando la distribución posterior, definimos el conjunto seguro un tiempo t y para un contexto dado ω_t como:

$$S_t = \{x \in \mathcal{X} \mid \mu_{t-1}^c(\omega_t, x) + \beta_t \sigma_{t-1}^c(\omega_t, x) \leq P_{\max}\}. \quad (5)$$

Los controles se seleccionan en cada periodo t utilizando la política CGP-UCB restringida al conjunto seguro:

$$x_t = argmax_{x \in S_t} \ \mu_{t-1}^f(\omega_t, x) + \sqrt{\beta_t} \sigma_{t-1}^f(\omega_t, x), \quad (6)$$

donde $\left(\sigma_t^f(z)\right)^2 = k_t^f(z, z)$.

Resumimos nuestro enfoque, denominado SBP-vRAN (Safe Bayesian optimization for Power consumption in vRANs), en el Algoritmo 2. Vale la pena mencionar que, en muchos escenarios prácticos, es deseable tener una restricción suave (soft constraint) en lugar de una restricción dura (hard constraint). Por ejemplo, puede interesarnos violar la restricción suave (aumentar el consumo de energía) para evitar un mal rendimiento del usuario. Ofrecemos dos alternativas para manejar este escenario. En primer lugar, podemos utilizar BP-vRAN diseñando $B(\cdot)$ de forma que un consumo de energía que supere la restricción incurra en un elevado coste monetario. Este enfoque proporciona garantías suaves en las que la restricción de potencia se cumplirá en promedio, pero no en todos los intervalos.

Como alternativa, podemos modificar la definición del conjunto seguro en la Ecuación (5). Así, podemos añadir una excepción tal que si el rendimiento esperado de todas las acciones del conjunto seguro está por debajo de un umbral de rendimiento r_{min}, incluya al menos una acción cuyo rendimiento esperado sea superior a $r_{\min}$. Utilizando este mecanismo, podemos establecer un requisito mínimo de rendimiento para la operación de la vBS.

Algoritmo 2. SBP-vRAN: Optimización online segura

1: Inputs: Espacio de control $\mathcal{X}$, kernel k, β, S_0, $P_{\max}$

2: Inicializar ${y^f}_0$, ${y^c}_0$, Z_0 como sets vacíos.

3: **For** $t = 1,2, \ldots$ **do**

4: Observar el contexto ω_t

5: Computar μ^f_{t-1}, σ^f_{t-1}, μ^c_{t-1} y σ^c_{t-1} usando ecuaciones (1)-(2)

6: $S_t = S_0 \cup \{x \in \mathcal{X} \mid \mu^c_{t-1}(\omega_t, x) + \beta_t \sigma^c_{t-1}(\omega_t, x) \leq P_{max}\}$

7: $x_t = argmax_{x \in \mathrm{S_t}}\, \mu^f_{t-1}(\omega_t, x) + \sqrt{\beta_t}\sigma^f_{t-1}(\omega_t, x)$

8: Medir $R^{dl}_t\left(\omega^{dl}_t, x^{dl}_t\right)$, $R^{ul}_t\left(\omega^{ul}_t, x^{ul}_t\right)$ y $P_t(\omega_t, x_t)$ al final del periodo de decisión t.

9: Computar $r_t(\omega_t, x_t)$

10: Actualizar $Z_t \leftarrow Z_{t-1} \cup z_t \coloneqq [\omega_t, x_t]$

11: Actualizar $y^f_t \leftarrow y^f_{t-1} \cup r_t(\omega_t, x_t)$

12: Actualizar $y^c_t \leftarrow y^c_{t-1} \cup P_t(\omega_t, x_t)$

End for

Convergencia de SBP-vRAN. Nótese que SBP-vRAN no expande explícitamente el conjunto seguro, como en otros trabajos como [16],[19]. En general, se necesita una expansión explícita del conjunto seguro (por ejemplo, mediante la exploración de los controles en el límite) para converger al verdadero conjunto seguro y, por tanto, alcanzar el control seguro óptimo. Sin embargo, hemos descubierto que nuestra función de adquisición puede maximizar el rendimiento y ampliar el conjunto seguro al mismo tiempo en algunas condiciones.

Supongamos que la función objetivo y la función restringida son suaves y están correlacionadas positivamente. En este caso, la maximización de la función objetivo también implica la ampliación del conjunto seguro. De hecho, la configuración óptima se sitúa en el límite del espacio de restricciones. Se trata de una suposición razonable en la práctica, como podemos evaluar empíricamente: Por un lado, la **Error! Reference source not found.** (a) muestra el caudal de enlace ascendente de nuestro vBS en función del esquema de codificación y modulación (MCS) y del tiempo de aire (dos de nuestras acciones de control). A partir de esta figura, podemos ver que, cuanto mayor sea el MCS y el tiempo de aire, mayor será el rendimiento. Por otro lado, la **Error! Reference source not found.** (b) muestra la potencia consumida en función de las mismas variables. Obsérvese que ambas figuras muestran la misma tendencia: a mayor rendimiento, mayor potencia consumida.

Hay que señalar que en este ejemplo sólo hemos considerado dos controles vBS (MCS y airtime). Sin embargo, aunque el comportamiento de la potencia se vuelve no lineal al incluir todas las dimensiones del problema, estas conclusiones también se mantienen en el problema completo. Es evidente que un mayor airtime proporciona un mayor rendimiento.

También es evidente que los MCS más altos proporcionan un mayor rendimiento en condiciones viables (SNR adecuada), ya que permiten empaquetar más símbolos de datos por unidad de tiempo. Del mismo modo, los MCS más altos conllevan un mayor consumo de energía, ya que el número de cálculos requeridos por los algoritmos de descodificación aumenta linealmente con el número de bits a descodificar. Además, una mayor potencia de transmisión permite mayores MCS y, por tanto, un mayor rendimiento. Por lo tanto, un mayor rendimiento suele ir asociado a un mayor consumo de energía.

Las anotaciones de la **Error! Reference source not found.** (a)-(b) ejemplifican cómo SBP-vRAN amplía el conjunto seguro. El conjunto seguro inicial (S_0) es un conjunto de configuraciones con el menor consumo de energía, es decir, MCS y airtime bajos. Este conjunto seguro inicial conservador evita violar la restricción desde el principio, pero también aumenta el tiempo de convergencia. El objetivo de SBP-vRAN es maximizar la función de recompensa r, que está directamente relacionada con el rendimiento y se define en SORUS-RAN-A1.2. Además, nuestra función de adquisición en la Ecuación (6) seleccionará controles con alto rendimiento pero también con alta incertidumbre. Estas condiciones las cumplen los controles en el límite del conjunto seguro. Al explorar estos controles estamos reduciendo la incertidumbre de su vecindad y, por tanto, ampliando el conjunto seguro. Después de unas pocas iteraciones ($t = n_1$), el conjunto seguro S_{n_1} se ha ampliado y el algoritmo puede ahora seleccionar configuraciones con mayor rendimiento. En ese momento, el algoritmo continuará explorando el límite de la restricción, ya que contiene las configuraciones con mayor rendimiento y también alta incertidumbre. Tras unas pocas iteraciones más, el conjunto seguro alcanzará el límite de la restricción, finalizando su expansión: las configuraciones óptimas caen en el límite del espacio de la restricción. Esto se demuestra más adelante en la evaluación experimental del algoritmo.

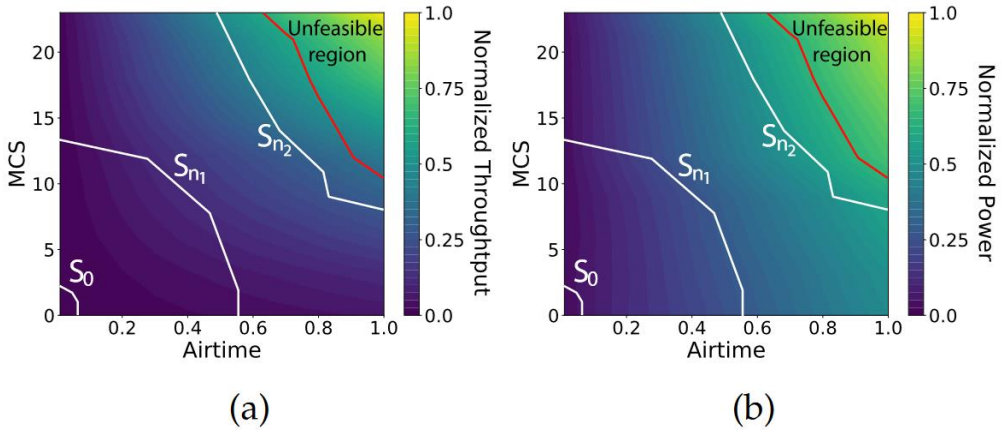

Figura 1. **Ejemplo de expansión del conjunto seguro en los dominios de rendimiento y potencia del enlace ascendente en función de dos variables de decisión: MCS del enlace ascendente y tiempo de emisión del enlace ascendente. A medida que SBP-vRAN explora, el conjunto seguro inicial S_0 se expande hasta alcanzar el límite de la región inviable, donde se encuentra el óptimo**

3.3. Caso de uso 3

En base a la formulación del problema presentada en el entregable SORUS-RAN-A1.2, necesitamos considerar dos restricciones para este caso de uso. En particular, las restricciones definen el máximo retardo en el servicio d^{max} y el valor mínimo de mAP (mean average precisión) ρ^{min}. En consecuencia, este algoritmo usará tres procesos gaussianos, uno para la función de coste y dos para las respectivas restricciones.

Los conjuntos de observaciones de las funciones de coste y de restricción en los puntos $Z_T = [z_1, \dots, z_T]$ hasta el período de tiempo T se denotan por $y_T^{(0)} = [u_1, \dots, u_T]$ (función de coste), $y_T^{(1)} = [d_1, \dots, d_T]$ (restricción del retardo máximo de servicio), $y_T^{(2)} = [\rho_1, \dots, \rho_T]$ (restricción del mAP), respectivamente. Así, usando las ecuaciones (1)-(2), obtenemos $\mu_T^{(i)}$, $\sigma_T^{(i)}$ para cada uno de los respectivos $y_T^{(i)}$, con $i = 0, 1, 2$.

Finalmente, definimos el conjunto seguro para este caso de uso teniendo en cuenta las dos restricciones:

$$S_t = S_0 \cup \{ x \in \mathcal{X} \mid \mu_{t-1}^{(1)}(c_t, x) + \beta\sigma_{t-1}^{(1)}(c_t, x) \leq d^{max} \wedge \mu_{t-1}^{(2)}(c_t, x) - \beta\sigma_{t-1}^{(2)}(c_t, x) \geq \rho^{min} \}. \quad (7)$$

El Algoritmo 3 resume todo la propuesta para este caso de uso. Al principio del periodo de decisión t, se observa el contexto c_t (línea 4). Basándose en el contexto observado c_t y en los vectores Z_{t-1} y $y_{t-1}^{(i)}$ del periodo de tiempo anterior, se calcula la distribución posterior de todas las funciones utilizando las ecuaciones (1)-(2) (línea 5). Tenga en cuenta que cuando no tenemos observaciones (Z_0 $y_0^{(i)}$, $\forall i$ vacíos) la distribución posterior es igual a la distribución a priori. Utilizando la expectativa y la incertidumbre de las funciones de restricción y se construye el conjunto seguro S_t usando la Ecuación (7) (línea 6). El control x_t se selecciona del conjunto seguro S_t basándose en la distribución posterior de la función de coste y la función de adquisición (línea 7). Al final del periodo de tiempo t, se observan todos los indicadores de rendimiento. A continuación, se computa el coste como se define en el entregable SORUS-RAN-A1.2 (línea 9). Por último, el nuevo par contexto-control z_t, el valor de la función de coste $u_t(c_t, x_t)$ y el valor de las funciones de restricción ($d_t(c_t, x_t)$ y $p_t(c_t, x_t)$) se añaden a sus respectivos vectores para generar Z_t y $y_t^{(i)} \forall i$ (líneas 10-13).

Algoritmo 3: Edge Bayesian Online Learning
1: Inputs: Espacio de control $\mathcal{X}$, kernel k, β, δ_1, δ_2, ρ^{min}, d^{max}
2: Inicializar $y_0^{(i)}$ $\forall i$ Z_0 como sets vacíos.
3: For $t = 1,2, \dots$ do
4: Observar el contexto c_t
5: Computar $\mu_{t-1}^{(i)}$, $\sigma_{t-1}^{(i)}$ $\forall i$, usando ecuaciones (1)-(2)
6: Estimar el conjunto seguro $S_t = S_0 \cup \{ x \in \mathcal{X} \mid \mu_{t-1}^{(1)}(c_t, x) + \beta\sigma_{t-1}^{(1)}(c_t, x) \leq d^{max} \wedge \mu_{t-1}^{(2)}(c_t, x) - \beta\sigma_{t-1}^{(2)}(c_t, x) \geq \rho^{min} \}$
7: $x_t = argmax_{x \in S_t} \mu_{t-1}^{(0)}(\omega_t, x) + \sqrt{\beta_t}\sigma_{t-1}^{(0)}(\omega_t, x)$

8: Medir $d_t(c_t, x_t)$, $\rho_t(c_t, x_t)$, $p_t^s(c_t, x_t)$, and $p_t^b(c_t, x_t)$ al final del periodo de decisión t

9: Computar el coste $u_t(c_t, x_t) = \delta_1 p_t^s(c, x) + \delta_2 p_t^b(c, x)$

10: Actualizar $Z_t \leftarrow Z_{t-1} \cup z_t \coloneqq [c_t, x_t]$

11: Actualizar $y_t^{(0)} \leftarrow y_{t-1} \cup u_t(\omega_t, x_t)$

12: Actualizar $y_t^{(1)} \leftarrow y_{t-1} \cup d_t(\omega_t, x_t)$

13: Actualizar $y_t^{(2)} \leftarrow y_{t-1} \cup \rho_t(\omega_t, x_t)$

End for

4 EVALUACIÓN EXPERIMENTAL

4.1. Casos de uso 1 & 2

En primer lugar, se van a evaluar experimentalmente los algoritmos de los casos de uso 1 y 2. Para ello, se usa la plataforma experimental detallada en la Sección 3.1 del entregable SORUS-RAN-A2.1.

Para la evaluación, consideramos las dimensiones de los controles (definidos en SORUS-RAN-A1.2) $|P^{dl}| = 20$, $|M^{dl}| = 28$, $|M^{ul}| = 24$, y $|A^{dl}| = |A^{ul}| = 11$, y por tanto el tamaño del conjunto de control es $|\mathcal{X}| \approx 1.6 \cdot 10^6$. Obsérvese que, para un período de decisión de 10 segundos, necesitaríamos hasta 185 días para explorar una vez cada política de control en $\mathcal{X}$, lo que pone de manifiesto la necesidad de una estrategia de aprendizaje eficiente en cuanto a datos. Aunque el Lema 1 garantiza la convergencia y el regret sub-lineal en general, puede lograrse una convergencia más rápida con información específica del problema. Por lo tanto, y en línea con trabajos anteriores [16],[17], seleccionamos $\beta^{1/2} = 2.5$, que muestra un buen rendimiento en nuestra configuración. En el caso de BP-vRAN, configuramos $\delta = 20$ y establecemos los parámetros a y b en la función de penalización $B(\cdot)$ definida en el entregable SORUS-RAN-A1.2, para penalizar severamente los valores de consumo de energía cercanos a b o superiores. En concreto, fijamos $a = 2.5$ y evaluamos distintos valores de b. Por último, presentamos los resultados de 10 experimentos (como mínimo), en los que representamos los valores medios y los percentiles 10^{th} y 90^{th} (zonas sombreadas).

4.1.1. Evaluación de la convergencia

Comenzamos evaluando la convergencia de los casos de uso 1 y 2, BP-vRAN y SBP-vRAN. Para ello, consideramos el caso especial de un único contexto y observamos su rendimiento a lo largo del tiempo hasta que convergen a políticas óptimas. Seleccionamos un contexto con alta SNR = 35 dB (CQI = 15) en DL y UL, y altas demandas de tráfico (relativas a la capacidad de nuestro testbed) iguales a 25 y 20 Mbps para DL y UL, respectivamente. Figura 2 y Figura 3 muestran la evolución temporal de diferentes métricas para ambos algoritmos durante 150 periodos de orquestación.

Analicemos en primer lugar los resultados de BP-vRAN mostrados en la Figura 2. Observamos que el consumo de energía y, en consecuencia, el rendimiento, se reducen para valores más bajos de b, por ejemplo, hay un 12,5% de caída de energía y un 33,75% de caída de rendimiento entre $b = 25$ y $b = 16$. Esto es intuitivo porque la reducción de b induce requisitos de potencia más estrictos. Nótese que $b = 16$ sólo penaliza el rendimiento de DL. Esto se debe a que impone un requisito de potencia leve, y por lo tanto BP-vRAN. Sólo sacrifica la potencia de transmisión, lo que reduce la SNR DL y, por tanto, el rendimiento DL. Los valores más bajos de b obligan a BP-vRAN a sacrificar también el rendimiento UL.

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

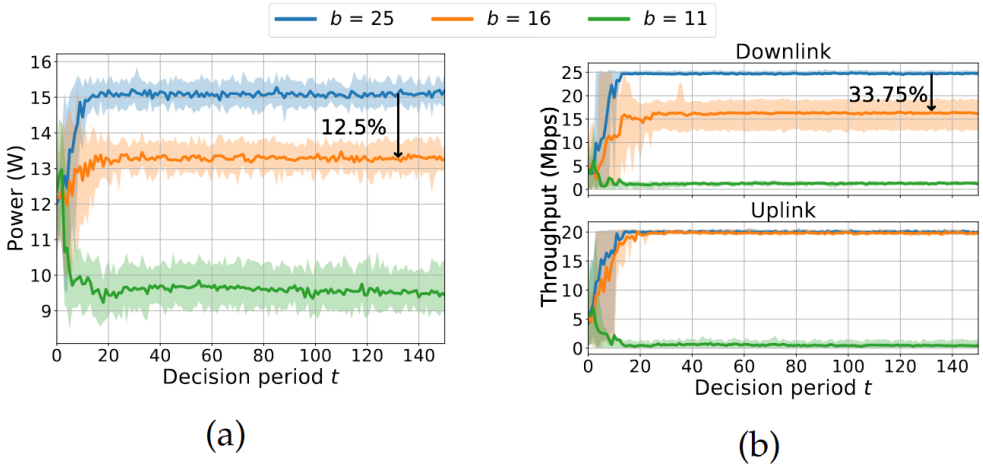

Figura 2. Evaluación de la tasa de convergencia de BP-vRAN para diferentes parámetros de la función objetivo

En cuanto a SBP-vRAN, evaluamos diferentes valores de $P_{\max}$ hasta $P_{\max} = 20$, que es un límite superior para el consumo de energía independientemente de la política y el contexto. Los resultados, en la Figura 3, muestran cómo SBP-vRAN aprende a utilizar configuraciones cumpliendo la restricción de energía con alta probabilidad, sacrificando el rendimiento cuando es necesario. Obsérvese que, en todos los casos, SBP-vRAN siempre selecciona políticas muy cercanas a $P_{\max}$. Esto se debe a que la política óptima, es decir, la que maximiza el rendimiento, suele requerir el máximo de potencia permitida $P_{\max}$. Para ello, SBP-vRAN expande gradualmente su conjunto seguro cerca de $P_{\max}$ y, por tanto, no es necesaria una estrategia explícita para expandir el conjunto seguro. En concreto, la Figura 4 muestra que todos los controles son seguros para $P_{\max} = 20$, con un 15,4% y un 53,2% menos de políticas seguras para $P_{\max} = 14$ y $P_{\max} = 12$, respectivamente. Como era de esperar, los valores más bajos de $P_{\max}$ incurren en un conjunto de políticas seguras más pequeño.

Concluimos esta evaluación con la observación de que, a pesar de utilizar un gran conjunto de políticas $\mathcal{X}$, ambos algoritmos convergen en 30 periodos de orquestación. Esto pone de relieve la eficiencia de datos de nuestras soluciones, que disciernen las políticas óptimas observando sólo un pequeño subconjunto de $\mathcal{X}$.

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

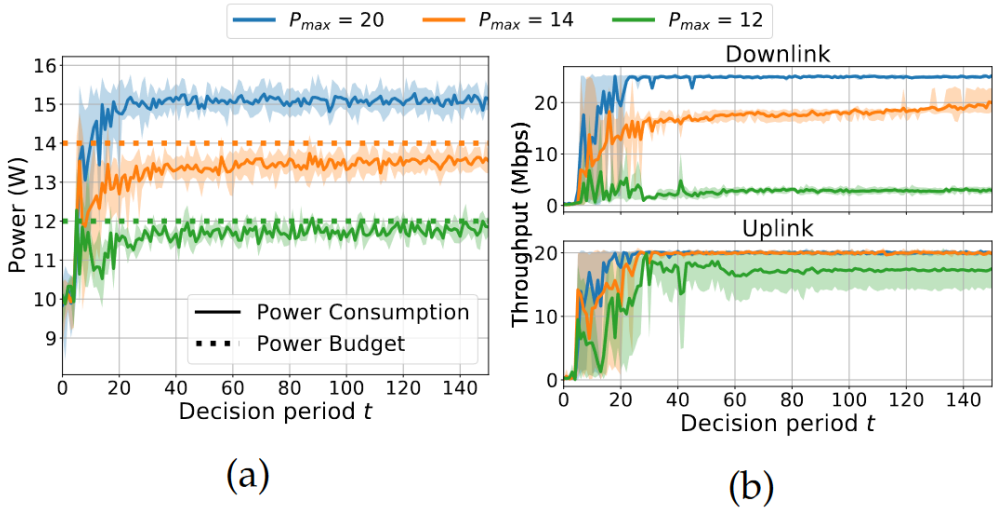

Figura 3. Evaluación de la tasa de convergencia de SBP-vRAN para distintos valores de P_{max}

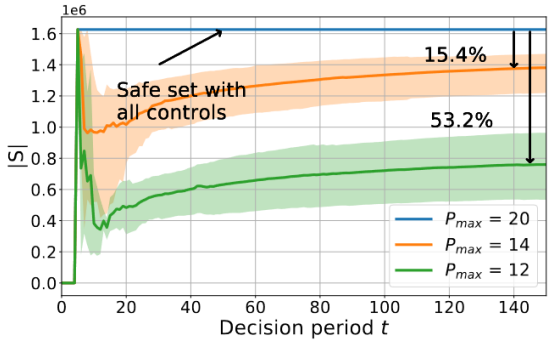

Figura 4. Evolución temporal del tamaño del conjunto seguro de SBP-vRAN para diferentes valores de P_{max}

4.1.2. Evaluación en contextos de red reales

A continuación, evaluamos el rendimiento de BP-vRAN y SBP-vRAN utilizando un patrón de tráfico realista de un día de [21] (Figura 5, parte superior). En cuanto a la calidad del canal, consideramos el peor de los casos, emulando a los UE con alta movilidad (Figura 5, parte inferior), lo que compromete la capacidad de la red (muy por debajo de la demanda). Debido a la granularidad de nuestro conjunto de datos de tráfico, en estos experimentos fijamos la duración del periodo de orquestación en 5 minutos (nótese que no hay pérdida de generalidad). Ejecutamos nuestros algoritmos durante dos días y presentamos los resultados del segundo día para centrarnos en el rendimiento alcanzado por el sistema. Su convergencia, evaluada en la subsección anterior, tarda sólo unos periodos. Esto es posible porque las políticas seleccionadas para contextos correlacionados también lo están, es decir, los conocimientos adquiridos para un contexto se transfieren a otros contextos similares. Por lo tanto, tras unas pocas iteraciones, los algoritmos seleccionan políticas eficientes incluso para contextos que no se han visto.

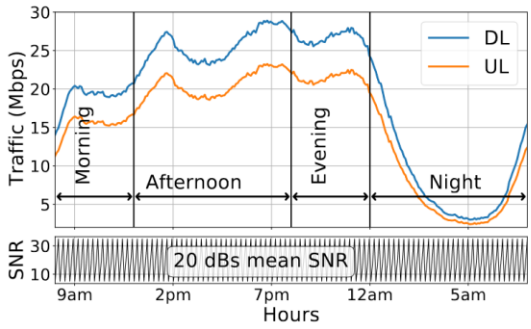

Figura 5. Patrón de tráfico de un día (parte superior) y patrón de calidad del canal (parte inferior)

Para eliminar el ruido introducido por la alta variabilidad de la SNR evaluada, cada punto de las Figura 6 y Figura 7 corresponde a la media de todos los puntos de un ciclo SNR, véase la Figura 5 en la parte de abajo. La Figura 6 muestra el consumo total de energía (a) y la evolución del rendimiento a lo largo del día (b) utilizando BP-vRAN y diferentes configuraciones de la función objetivo. Observamos que el consumo de energía evoluciona con la demanda de tráfico y con el valor seleccionado de b. Por ejemplo, cuando $b = 16$, el rendimiento alcanzado se ve penalizado a favor de un mejor consumo de energía durante el día, pero no se produce ninguna degradación del rendimiento durante la noche (entre las 2 y las 7 de la mañana). De forma similar, la Figura 7 muestra el rendimiento de SBP-vRAN en los mismos escenarios. En concreto, SBP-vRAN logra satisfacer la restricción de energía con probabilidades 0.99 y 0.93 cuando $P_{\max}$ es igual a 14 y 12, respectivamente, al tiempo que maximiza el rendimiento (que se calculó mediante búsqueda exhaustiva).

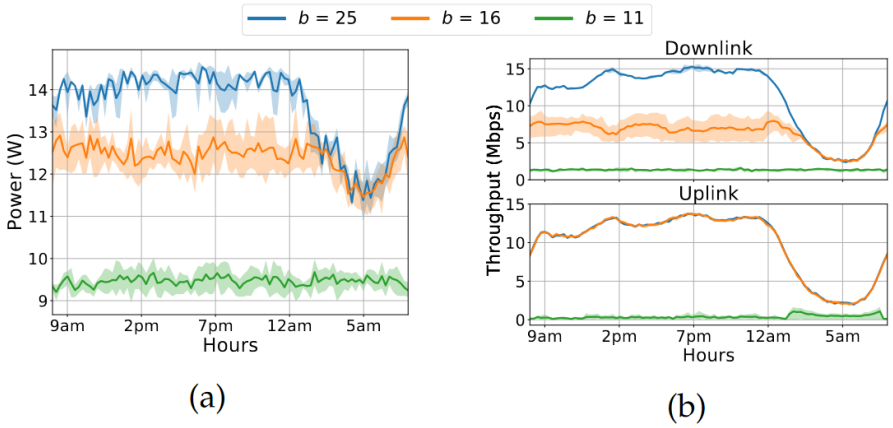

Figura 6. Evaluación del rendimiento de BP-vRAN a lo largo de un día

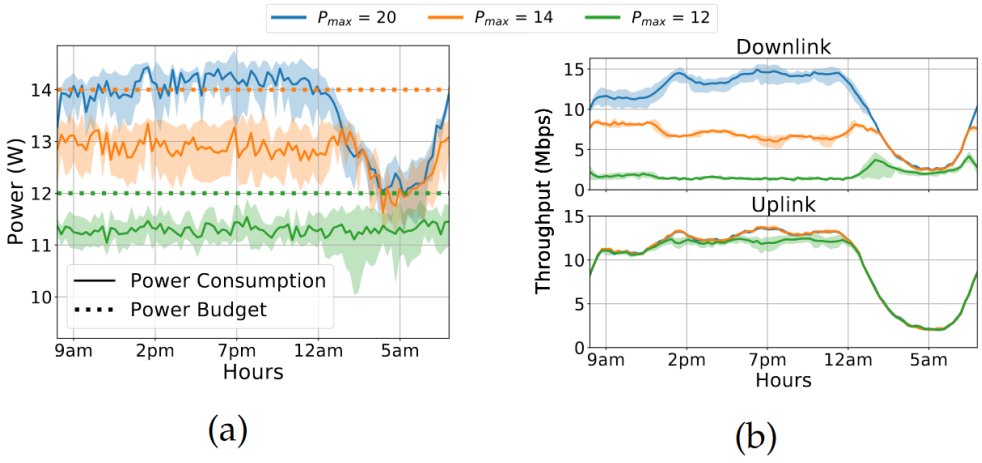

Figura 7. Evaluación del rendimiento de SBP-vRAN a lo largo de un día

4.1.3. Comparación con otros métodos

Completamos nuestra evaluación comparando nuestras soluciones con un algoritmo de aprendizaje profundo por refuerzo de última generación: el denominado DDPG. Este algoritmo necesita ser personalizado ya que está diseñado para resolver el problema full-RL mientras que en este trabajo nos enfrentamos a un problema contextual bandit. Existen dos diferencias principales entre estos dos problemas. En primer lugar, el full-RL considera que las acciones seleccionadas (políticas de control) tienen un impacto en los estados futuros (contextos). Este supuesto no se cumple en nuestro caso, ya que la configuración del vBS no afecta a los contextos futuros (carga de tráfico y calidad del canal de los usuarios). En segundo lugar, en el problema full-RL, la recompensa puede retrasarse en el tiempo, mientras que en nuestro escenario el rendimiento está disponible al final del periodo de decisión.

El DDPG se implementa utilizando una arquitectura de NN profunda actor-critic y, para adaptarla al problema de bandits contextuales, configuramos la NN del critic para aproximar la función de recompensa en lugar de la función de valor Q (véase [5] para más detalles). Consideramos la misma arquitectura de NN que en [5] pero utilizamos una sigmoidea como función de activación de la capa de salida de la NN del actor. Dado que el espacio de acción del DDPG es continuo (la salida del actor es un vector continuo con las mismas dimensiones que $\mathcal{X}$), las acciones seleccionadas se ajustan a las políticas de control más cercanas que pueden ser configuradas por el vBS. Además, optimizamos los hiperparámetros para minimizar el tiempo de convergencia. Nuestros experimentos muestran que el DDPG converge a las mismas soluciones que los algoritmos bayesianos propuestos, pero carece de velocidad de convergencia y versatilidad.

Para el caso de uso, configuramos la función de recompensa del DDPG para que sea la función objetivo. Figura 8 muestra la evolución temporal de la función objetivo para BP-vRAN y DDPG, para diferentes valores de b. En particular, DDPG converge a la misma política óptima aprendida por BP-vRAN, pero tiene que invertir un orden de magnitud más de tiempo. La razón principal de esta diferencia es que nuestro enfoque infiere correlaciones en la función objetivo sobre el espacio contexto-acción de manera más eficiente; y por lo tanto encuentra políticas óptimas, incluso para los pares contexto-acción no vistos. Esto resalta la eficiencia en el uso de los datos de la solución

basada en GP. También vale la pena recordar que, a diferencia de nuestro punto de referencia, BP-vRAN tiene garantías matemáticas en el rendimiento como se desarrolla en este entregable.

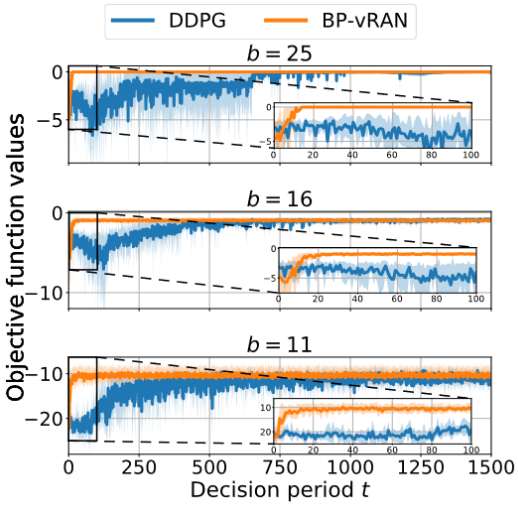

Figura 8. Comparación de BP-vRAN con DDPG modificado

Con el fin de implementar el segundo caso de uso, consideramos una función de recompensa personalizada para el DDPG. La recompensa se codifica mediante una función escalón que toma el valor de la función objetivo cuando la potencia observada es inferior a $P_{\max}$, y el valor mínimo de recompensa en caso contrario. La Figura 9 muestra la evolución en el tiempo del consumo de energía y el rendimiento asociado de la vBS para SBP-vRAN y DDPG. Comenzamos el experimento fijando la restricción de potencia en 15 W y cambiándola a 13 W en el periodo de decisión $t = 2000$.

Nuestros resultados arrojan tres observaciones: *(i)* SBP-vRAN logra mejoras considerables de convergencia sobre su punto de referencia (aproximadamente, un orden de magnitud). *(ii)* SBP-vRAN no se ve afectado por un cambio repentino en la restricción de potencia; tenga en cuenta que sólo requiere el cambio de $P_{\max}$ en la línea 6, Algoritmo 2. Por el contrario, DDPG necesita cambiar la configuración de la función de recompensa, lo que obliga a reiniciar su proceso de aprendizaje desde cero, fallando la restricción dura hasta el período de decisión 3500, aproximadamente. *(iii)* DDPG no puede realizar una exploración segura: debe utilizar políticas que violen la restricción de potencia para aprender donde está el límite de la restricción. Por otro lado, nuestro enfoque calcula la incertidumbre de cada estimación, lo que nos permite implementar la exploración segura y satisfacer la restricción con alta probabilidad. *(iv)* Aunque el DDPG puede potencialmente encontrar mejores soluciones debido a su espacio de acción continua, nuestros resultados muestran que ambos enfoques convergen a la misma solución debido a la discretización de grano fino del espacio de acción de BP-vRAN y SBP-vRAN.

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

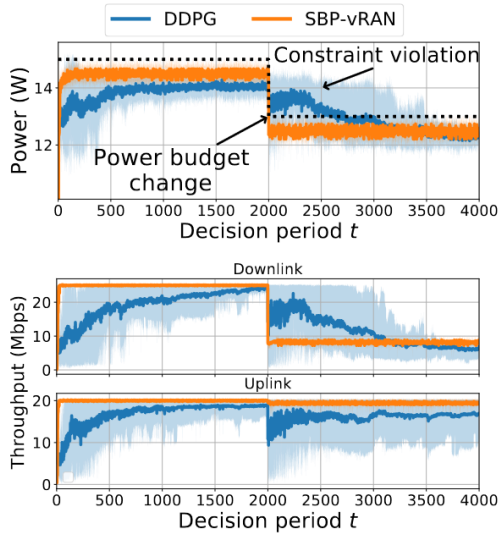

Figura 9. Comparación de SBP-vRAN con DDPG modificado

Por último, es importante señalar que el inconveniente inherente de los métodos basados en GP es la complejidad de cálculo (para la descomposición de Cholesky) en cada periodo de orquestación, donde N es el número de puntos de datos. Sin embargo, en nuestros experimentos observamos que la velocidad de convergencia sin precedentes de estos métodos se amortiza en muy poco tiempo. Además, comprobamos que estos cálculos no inducen retrasos, ya que, según las especificaciones de la O-RAN, existe una ventana de tiempo suficientemente amplia para actualizar la política.

4.2. Caso de uso 3

En esta sección se evalúan experimentalmente el algoritmo del caso de uso 3. Para ello, se usa la plataforma experimental detallada en la sección 6.1 del entregable SORUS-RAN-A2.1.

Consideramos las dimensiones de los controles (definidos en SORUS-RAN A1.2) $|H| = |A| = |\Gamma| = |M| = 11$; por lo tanto, hay un gran número de $|\mathcal{X}| = 11^4 \approx 14.6 \cdot 10^3$políticas de control, que, en combinación con el efecto de los posibles contextos, pone de relieve la necesidad de un mecanismo de aprendizaje eficiente de datos. Dada la complejidad de ejecutar experimentos con múltiples usuarios, nos basamos en un solo usuario en la mayoría de nuestros experimentos (que hacen triviales los controladores de capas bajas). Sin embargo, cuando es necesario (al evaluar múltiples usuarios heterogéneos), adoptamos controladores sencillos (por ejemplo, la programación de la capa MAC) que se detallan cuando es pertinente.

En línea con trabajos anteriores [16][17], seleccionamos $\beta^{1/2} = 2.5$, que muestra un buen rendimiento en nuestras evaluaciones. Configuramos la duración de los periodos de tiempo en 2 segundos. Por último, a menos que se indique lo contrario, trazaremos nuestros resultados con líneas y áreas sombreadas que representan, respectivamente, el valor mediano y los percentiles 10^{th} y 90^{th}, a lo largo de 10 repeticiones independientes.

4.2.1. Evaluación de la convergencia

Para evaluar la convergencia del algoritmo, consideramos un único contexto y un determinado conjunto de restricciones con ρ^{min} (rendimiento mAP mínimo) y d^{max} (retardo de servicio máximo). Posteriormente se evalúan los cambios dinámicos de contexto y las distintas restricciones. En concreto, fijamos la SNR media en 35 dB (buenas condiciones inalámbricas), $\delta_1 = 1$ mu/W, $\rho^{min} = 0.5$, y $d^{max} = 0.4$s. Figura 10 representa la evolución en el tiempo del coste (u_t), el rendimiento mAP (ρ_t), el retardo (d_t) y el consumo de energía del servidor y la BS (p_t^s y p_t^b) en función de δ_2. La primera observación es que el coste u_t (gráfico superior) converge en unos 25 periodos en todos los $\delta_2 = \{1, 2, 4, 8, 32\}$. Los valores más altos de δ_2 inducen un mayor coste a medida que crece el precio del Watt de la BS. Sorprendentemente, tanto el rendimiento mAP como el retardo caen dentro de las restricciones del sistema seleccionado en la convergencia con alta probabilidad. De hecho, observamos resultados coherentes (velocidad de convergencia, satisfacción de las restricciones del sistema) independientemente del contexto y de las restricciones del sistema.

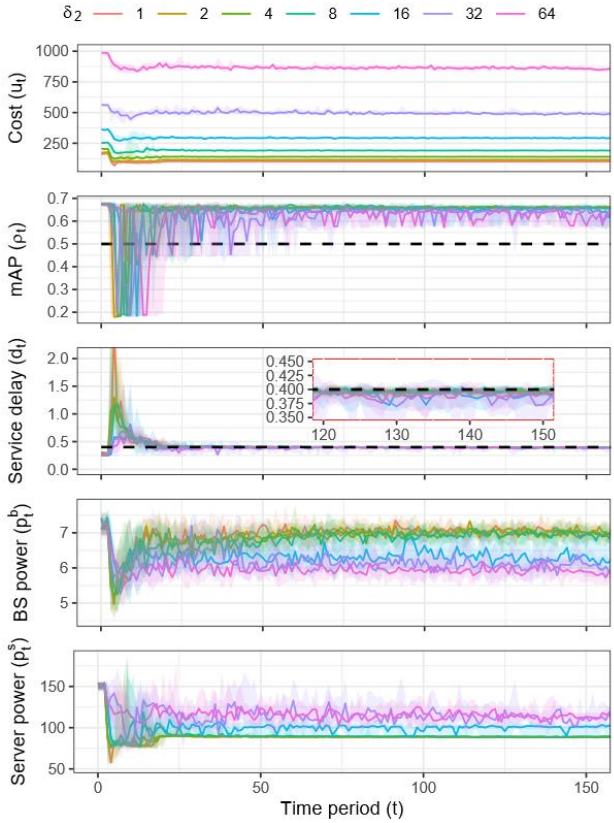

Figura 10. Evaluación de la convergencia. Escenario con condiciones de canal estables (sin cambios de contexto), $\delta_1 = 1$ mu/W, $\rho^{min} = 0.5$, y $d^{max} = 0.4$s. La línea negra discontinua indica la restricción

El consumo de energía del sistema presenta interesantes compensaciones con δ_2. Un δ_2 pequeño (por ejemplo, 1) induce un consumo elevado en la BS, pero bajo en el servidor. Esto se debe a que el consumo de energía neto máximo de nuestra BS (alrededor de 7.25 W) es mucho menor que el del servidor (entre 85 y 180 W). Por lo tanto, si el coste asociado (mu/W) es similar tanto para el vBS como para el servidor, el algoritmo minimizará el consumo de energía del servidor a expensas de un pequeño peaje energético de la vBS. Sin embargo, cuando δ_1 es relativamente alto (por ejemplo, 64), el coste real asociado a la huella energética de la vBS llega a ser comparable, o incluso

superior, al del servidor. Por lo tanto, nuestra propuesta lleva al sistema a configuraciones que minimizan el consumo de energía de la vBS a expensas de más energía del servidor. Este último caso es relevante para situaciones en las que una small cell tiene un presupuesto de energía estricto o restricciones de refrigeración. De hecho, los diferentes tipos de vBS tienen diferentes huellas energéticas, lo que motiva la necesidad de enfoques que aprendan la relación entre el consumo de energía, el rendimiento y las políticas de configuración.

4.2.2. Evaluación de escenarios estáticos

Echemos ahora un vistazo más de cerca el consumo de energía y las respectivas políticas del algoritmo para diferentes restricciones y valores de δ_2. La Figura 11 muestra el consumo de energía y el coste normalizado una vez que el algoritmo ha convergido para $\delta_1 = 1$ y $\delta_2 = \{1, 2, 4, 8, 16, 32, 64\}$ mu/W. Calculamos el coste normalizado de forma independiente para cada δ_2 de modo que podamos comparar entre diferentes valores de δ_2. Ahora probamos distintas configuraciones de las restricciones: *(i)* $d^{max} = 0.5$ s, $\rho^{min} = 0.4$ (restricciones laxas), *(ii)* $d^{max} = 0.4$ s, $\rho^{min} = 0.5$ (restricciones medias), y *(iii)* $d^{max} = 0.3$ s, $\rho^{min} = 0.6$ (restricciones estrictas), representadas en rojo, verde y azul en la figura. Además, representamos con líneas discontinuas el coste alcanzable mediante un oráculo computado offline, que obtuvimos utilizando un procedimiento de búsqueda exhaustiva sobre todo el espacio de control, algo que requiere de mucho tiempo. Aunque este enfoque es inviable en la práctica, es un buen punto de referencia para evaluar empíricamente la optimalidad de nuestra propuesta.

Ignorando, por ahora, las diferencias entre las distintas configuraciones de las restricciones (diferentes colores en los gráficos), podemos hacer dos observaciones. En primer lugar, podemos confirmar nuestra observación anterior de que los valores más altos de δ_2 (en comparación con δ_1) dirigen al algoritmo a desplazar el consumo de energía desde el servidor a la BS (y viceversa). En segundo lugar, el algoritmo propuesto es capaz de conducir el sistema a puntos cercanos al óptimo de funcionamiento, al comparar el coste del algoritmo con el obtenido por nuestro oráculo.

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

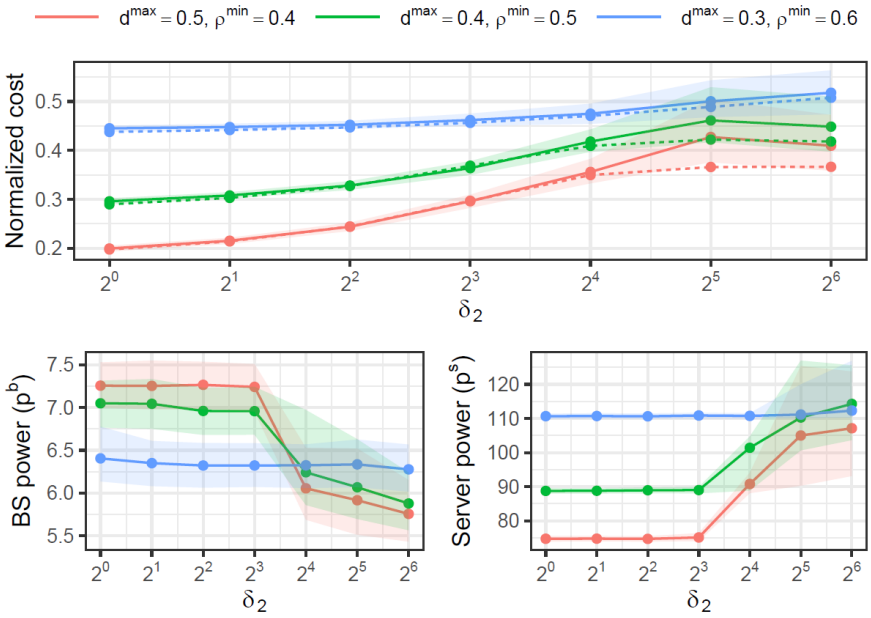

Figura 11. Consumo de energía y coste normalizado para un único contexto en función de δ_2, con $\delta_1 = 1$ mu/W. Las líneas discontinuas representan nuestro enfoque de búsqueda exhaustiva

Más en detalle, la figura muestra un comportamiento muy diferente según los distintos ajustes de las restricciones (colores en el gráfico). En el caso de $d^{max} = 0.5$ s, $\rho^{min} = 0.4$ (restricciones laxas, en rojo en el gráfico), se produce un cambio drástico en las políticas seleccionadas y el consumo de energía resultante a medida que aumentamos δ_2. Debido a que estos ajustes son bastante laxos, el algoritmo tiene más margen para explorar (y luego seleccionar una política de) un mayor espacio de políticas admitidas. Esto se hace evidente cuando comparamos su coste normalizado con el de los ajustes más estrictos ($d^{max} = 0.3$ s, $\rho^{min} = 0.6$, línea azul en la figura). Para $\delta_2 = 1$, el coste mínimo alcanzado por nuestro algoritmo es 25% mayor para este último, y 10% para $\delta_2 = 64$.

Además, el coste normalizado crece sistemáticamente, aunque con una diferencia cada vez menor entre los distintos ajustes de las restricciones, a medida que aumentamos δ_2. Esto ocurre porque, en nuestro banco de pruebas, el rango de valores de potencia que puede consumir la BS (en todas las políticas) oscila entre 4 y 8 W, que es sustancialmente menor que el del servidor (entre 50 y 200 W). Como resultado, cuando aumentamos δ_2, es decir, cuando aumentamos la importancia dada a reducir la potencia de la BS, la varianza de costes entre políticas se reduce. Huelga decir que esto puede ser diferente con distintos tipos de BS, como las macro celdas.

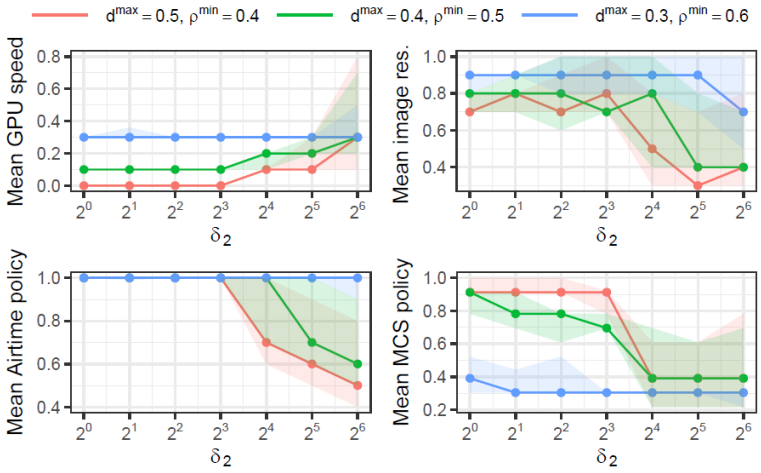

Figura 12. Políticas para un único contexto en función de δ_2, con $\delta_1 = 1$ mu/W

Por último, la Figura 12 muestra las políticas de control correspondientes para los mismos escenarios mostrados en la Figura 11. Veamos primero los ajustes laxos ($d^{max} = 0.5$ s, $\rho^{min} = 0.4$, representados con líneas rojas en la figura). Cuando δ_2 es pequeño, nuestra propuesta impone políticas de bajo consumo del lado del servidor, es decir, políticas de baja velocidad de la GPU. Esto sin duda ayuda a reducir el consumo del servidor y, en consecuencia, el coste global. Sin embargo, para cumplir con las restricciones de rendimiento, el algoritmo tiene que compensar las políticas de baja velocidad de la GPU con resoluciones de imagen más altas y políticas de radio más altas que facilitan el trabajo del servicio minimizando el retraso, lo que se produce a expensas de un mayor consumo de energía de la BS. Por el contrario, cuando δ_2 aumenta, el algoritmo selecciona políticas de radio de bajo consumo y, para compensar, resoluciones de imagen más bajas y políticas de velocidad de GPU más altas que ayudan a reducir el retardo del servicio. Por otro lado, para el escenario con las restricciones más estrictas ($d^{max} = 0.3$ s, $\rho^{min} = 0.6$, líneas azules), el algoritmo se ve obligado a tratar con un espacio más pequeño de políticas viables. Por lo tanto, todas las políticas son más o menos coherentes a través de diferentes valores de δ_2 (con ligeras diferencias para los ajustes más altos).

4.2.3. Usuarios heterogéneos

En un esfuerzo por reducir el problema de la dimensionalidad, agregamos estadísticas de usuarios individuales (SNR media, SNR varianza, etc.) al describir el contexto. Para validar que esta elección de diseño no compromete la optimalidad, hemos realizado una serie de experimentos con múltiples usuarios heterogéneos. Sin pérdida de generalidad, adoptamos un sencillo mecanismo de control de bajo nivel para hacer cumplir las políticas seleccionadas cuando se asignan recursos a usuarios individuales: *(i)* un enfoque de programación de radio Round-Robin en la capa MAC de la BS, *(ii)* igual resolución de imagen entre usuarios, *(iii)* enfoque de selección MCS legado de srsRAN [8] (limitado superiormente por la política), y *(iv)* la mayor velocidad de GPU para manejar fotogramas de vídeo individuales permitida por la política.

Entrenamos el algoritmo con un número variable de usuarios heterogéneos con calidad de canal cambiante. Una vez entrenado, evaluamos el rendimiento del algoritmo en escenarios con un número fijo de N usuarios heterogéneos. El primer usuario tiene las mejores condiciones de canal

(SNR = 30 dB de media) y cada usuario adicional tiene una SNR un 20% inferior. Elegimos trivialmente $d^{max} = 2$ s, $\rho^{min} = 0.6$ para que el sistema tenga una solución factible en el peor de los casos (con 6 usuarios). Figura 13 representa el coste medio del sistema y todos los indicadores de rendimiento para escenarios con diferentes valores de N. Hacemos esto para diferentes ponderaciones δ_2 en el compromiso entre el consumo de energía del servidor y el del vBS ($\delta_1 = 1$ en todos los escenarios). A su vez, la Figura 14 muestra las políticas seleccionadas en promedio.

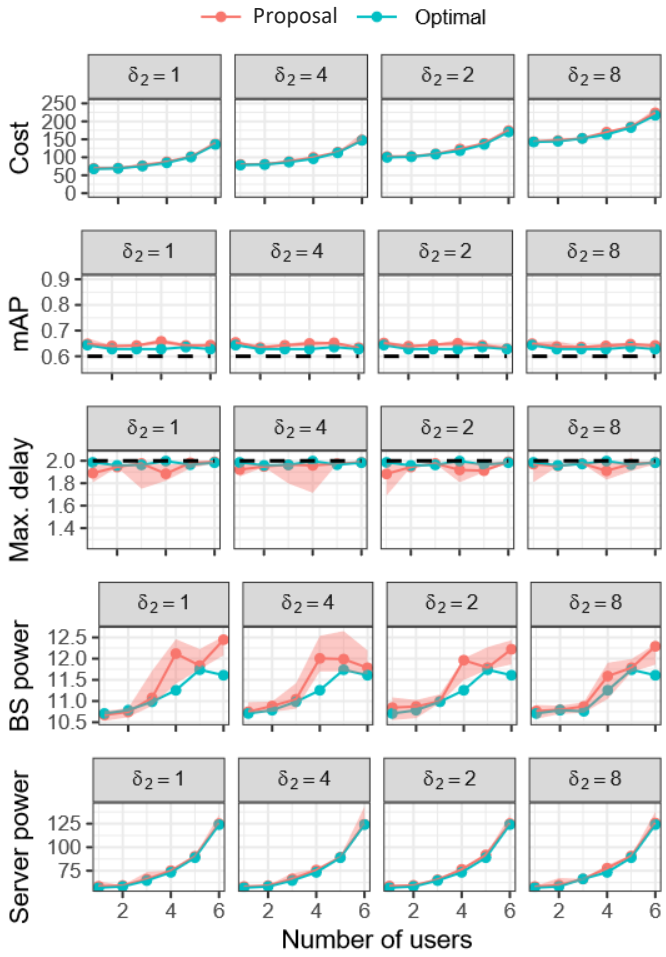

Figura 13. Brecha de optimalidad empírica en escenarios con múltiples usuarios heterogéneos. Cada escenario tiene N usuarios con diferentes condiciones de SNR: el usuario 1 tiene las mejores condiciones de canal (SNR = 30 dB de media) y cada usuario adicional tiene un 20% menos de SNR. Evaluamos distintos valores de δ_2. $\delta_1 = 1$. La línea negra discontinua indica la restricción de servicio

Comparamos el rendimiento del algoritmo propuesto con el de un oráculo óptimo que encuentra la mejor combinación posible de políticas offline tras una búsqueda exhaustiva en la que se conoce toda la dinámica del sistema. Por lo tanto, aunque es inviable para su uso en la práctica, proporciona un costo límite inferior que nos ayuda a evaluar la brecha de optimalidad del algoritmo empíricamente. Los resultados muestran que el rendimiento alcanzado por nuestra propuesta está muy cerca de la del oráculo, bien dentro de 2%. Además, el algoritmo satisface las restricciones de

servicio (representado en la Figura 13 con líneas oscuras discontinuas) con probabilidad 0.98. Esto confirma que las estadísticas agregadas de todos los usuarios bastan para ofrecer un buen rendimiento, manteniendo al mismo tiempo la complejidad del problema.

También podemos observar que el coste total aumenta con el número de usuarios. La razón es que, como cada usuario adicional tiene una SNR menor, su tiempo de transmisión es mayor. Como consecuencia, el algoritmo se ve obligado a invertir más recursos (es decir, airtime, velocidad GPU) en el sistema para compensar esta degradación de las condiciones inalámbricas medias.

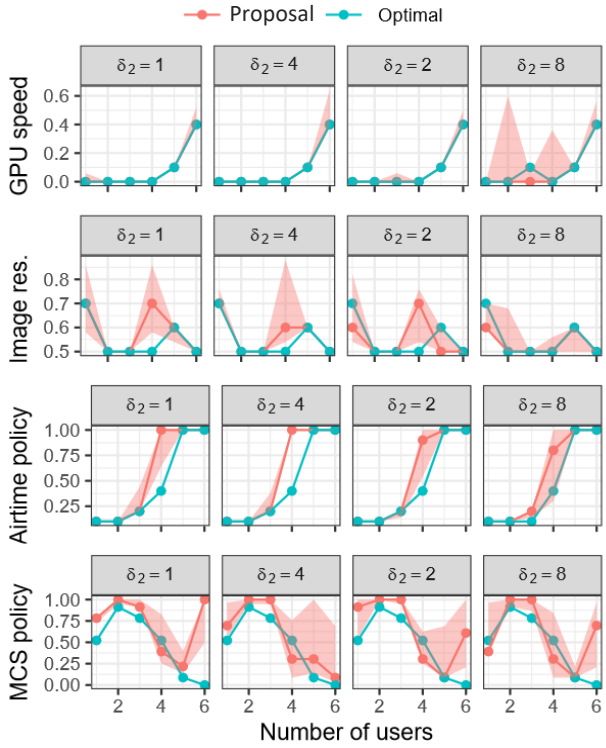

Figura 14. Políticas óptimas con múltiples usuarios heterogéneos. Cada escenario tiene N usuarios con diferentes condiciones de SNR: el usuario 1 tiene las mejores condiciones de canal (SNR = 30 dB de media) y cada usuario adicional tiene un 20% menos de SNR. Evaluamos distintos valores de δ_2. $\delta_1 = 1$

4.2.4. Escenarios dinámicos

Ahora probamos el rendimiento de nuestra propuesta en presencia de dinámicas de contexto rápidas y cambios de restricción repentinos. Comencemos por lo primero. Para ello, desplegamos el algoritmo sin entrenar en un entorno en el que las condiciones inalámbricas varían rápidamente entre 5 y 38 dB, como se muestra en el primer gráfico de la Figura 15, y establecemos $\delta_1 = 1$ y $\delta_2 = 8$. La parte superior del gráfico muestra el tamaño del conjunto de control seguro con el tiempo. Como era de esperar, el conjunto seguro se reduce rápidamente en aproximadamente 25 períodos de tiempo y luego se adapta a los cambios contextuales eventuales, con fluctuaciones que coinciden

con los cambios de contexto. Sorprendentemente, el algoritmo propuesto converge en sólo 3 ciclos en todos los contextos evaluados. Esto es posible porque el conocimiento adquirido por el algoritmo para un contexto es en realidad transferido a través de contextos similares. Es decir, es capaz de seleccionar políticas juiciosas, que se muestra en las parcelas restantes de la figura, incluso para contextos no vistos. En concreto, para esta elección de parámetros δ, la política de velocidad de la GPU y la política de MCS varían mucho en función de la dinámica del contexto, mientras que la resolución de la imagen y la política de tiempo de emisión se mantienen constantemente altas. Cabe mencionar que la dinámica de estas políticas es sustancialmente diferente para distintos valores de δ (no se muestra por motivos de espacio).

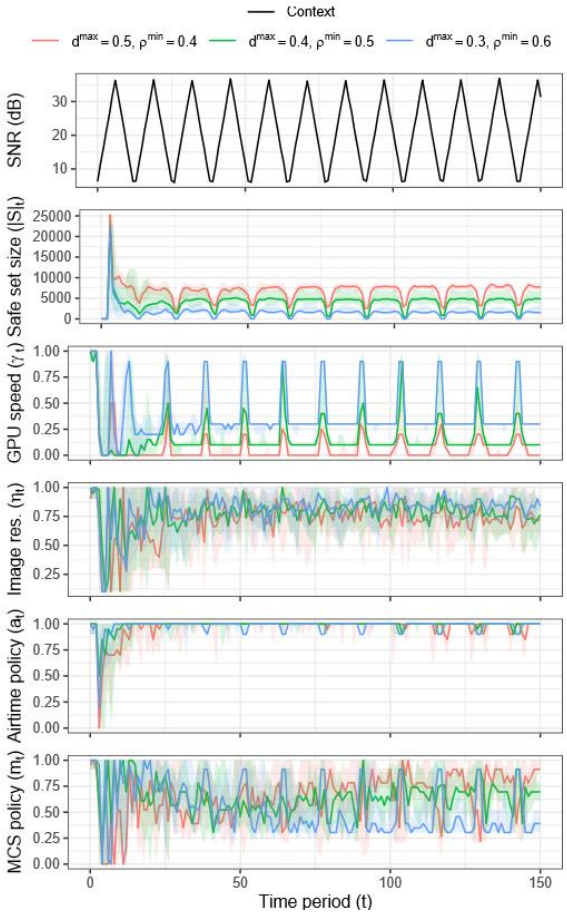

Figura 15. Evolución de las políticas para contextos dinámicos ($\delta = 8$)

Lo anterior ilustra una de las principales ventajas de nuestro enfoque, que toma decisiones apropiadas —incluso para contextos no vistos antes— infiriendo correlaciones entre la función de coste y el espacio de control del contexto. Los enfoques convencionales basados en NN son mucho menos eficientes en este sentido, lo que convierte a nuestro método en una solución especialmente eficiente en términos de datos.

Para evaluar esto, implementamos una versión personalizada del DDPG [5]. Este punto de referencia se inspira en [5], que es el trabajo más relacionado con el nuestro. Dado que el DDPG está diseñado para abordar el problema full-RL, necesitamos adaptarlo para abordar un problema de contextual bandit, de acuerdo con la formulación del problema detallada en SORUS-RAN-A1.2.

El algoritmo DDPG utiliza una arquitectura de NN actor-critic, pero el crítico, en lugar de aproximar la función Q (problema full-RL), aprende una nueva función de coste denominada *coste DDPG*. El *coste DDPG* toma el valor de la función de coste u cuando se satisfacen todas las restricciones, y el valor del coste máximo en caso contrario. Tenga en cuenta que el DDPG no maneja las restricciones de forma natural y mediante el uso de la función *coste DDPG* permitimos que el algoritmo lo haga.

El DDPG es particularmente atractivo para este tipo de problemas porque opera con espacios de control de valores continuos. Por el contrario, los enfoques basados en valores (como Deep Q-Networks) son poco prácticos para grandes espacios de control, como el nuestro [22]. Modificamos ligeramente la arquitectura presentada en [5] con una función sigmoidea para la salida del actor y optimizamos todos los hiperparámetros (como el decaimiento) para minimizar el tiempo de convergencia y el rendimiento.

A continuación, probamos el algoritmo propuesto y DDPG en un escenario dinámico en el que la configuración de las restricciones cambia con el tiempo: *(i)* $d^{max} = 0.5$ s, $\rho^{min} = 0.4$ desde $t = 0$ hasta $t = 1000$; *(ii)* $d^{max} = 0.4$ s, $\rho^{min} = 0.6$ desde $t = 1000$ hasta $t = 2000$; y *(iii)* $d^{max} = 0.5$ s, $\rho^{min} = 0.5$ a partir de $t = 2000$.

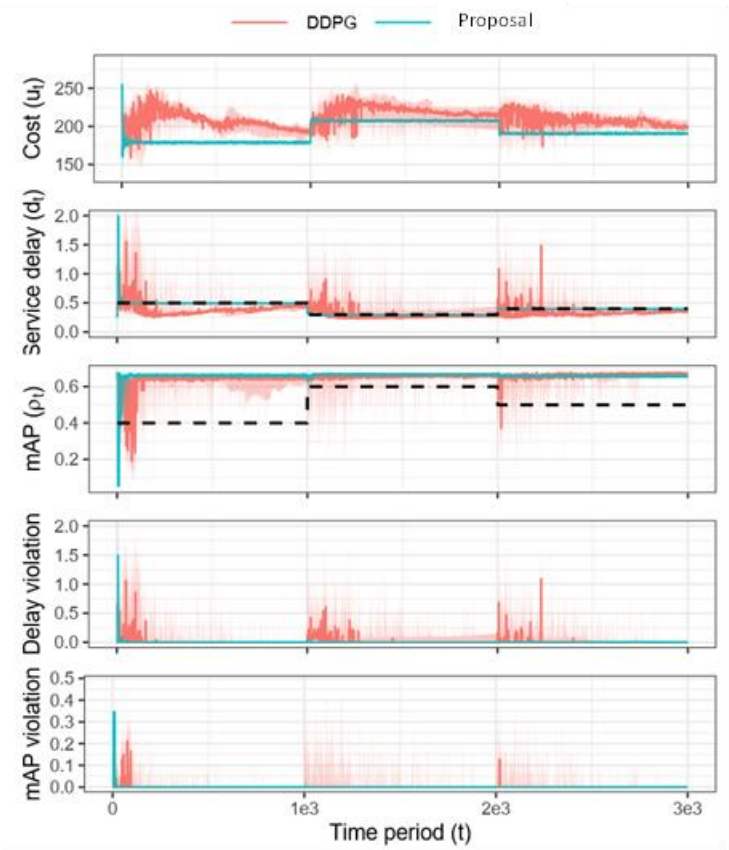

Figura 16. Evolución del retardo y del mAP tras cambios en la configuración de las restricciones para el algoritmo propuesto y un enfoque DDPG implementado con redes neuronales ($\delta = 8$)

La Figura 16 representa la evolución en el tiempo del retardo del servicio y el rendimiento mAP para ambos enfoques. No es sorprendente que nuestra propuesta converja rápidamente a políticas que respetan las restricciones de rendimiento, incluso cuando cambian repentinamente. La naturaleza

no paramétrica de nuestro enfoque y el hecho de que podemos calcular conjuntos de control seguro para cualquier ajuste restringido basado en datos anteriores permite a nuestra propuesta conducir el sistema a los nuevos puntos óptimos de operaciones casi instantáneamente. En marcado contraste, el punto de referencia basado en NNs tarda un número sustancialmente mayor de periodos de tiempo en encontrar el nuevo óptimo —de hecho, es incapaz de converger antes de los cambios de restricción— y no se adapta con gracia a los cambios de restricción porque las NN son modelos paramétricos que necesitan volver a aprender ante tales cambios.

5 CONCLUSIONES

Las soluciones actuales de políticas de control energético en las BS no permiten obtener un rendimiento óptimo de las vBS, debido a su complejidad, la cantidad de variables que influyen y la dependencia hardware. Por ello, existe la necesidad de implementar políticas en las vBS que permitan una rápida adaptación en escenarios de alta variabilidad, algo clave para reducir el consumo y, por tanto, reducir costes sin perjuicio de mantener la calidad de servicio en el Edge. Con este objetivo, se ha propuesto un framework de aprendizaje automático del perfil operativo de la vBS capaz de configurar la misma en base a las necesidades de la red, la disponibilidad de recursos y las restricciones energéticas presentes.

Este framework incluye diferentes algoritmos basados en la teoría de optimización bayesiana y GPs para modelar las vBS y el problema como un contextual bandit. Esta propuesta ha sido probada proponiendo tres casos de uso (definidos en detalle en el entregable SORUS-RAN-A1.2) enfocados en i) el equilibrio entre rendimiento y coste (algoritmo BP-vRAN), ii) imposición de restricciones estrictas de potencia (algoritmo SBP-vRAN) y iii) optimización conjunta de vBS y servicios Edge de IA (algoritmo Edge Bayesian Online Learning).

Respecto a los casos de uso 1 y 2, se ha comprobado, comparando con algoritmos de DDPG de última generación mediante experimentos de convergencia hacia la política óptima y de adaptación a cambios de restricciones que nuestra solución converge hacia puntos cercanos al óptimo de funcionamiento y se adapta a los cambios de restricciones de potencia hasta un orden de magnitud más rápido. Por último, en el caso de uso 3, se ha evaluado en escenarios estáticos, heterogéneos y dinámicos, comparando también con una implementación basada en DDPG. Se ha utilizado una plataforma experimental basada en srsRAN para inyectar trazas de tráfico real y medir el consumo de energía de la vBS. Se puede observar que nuestro framework explora las configuraciones ajustándose a las restricciones. Comparando nuestra solución con la basada en RL hemos podido comprobar que nuestra propuesta converge rápidamente a políticas que respetan las restricciones de rendimiento, incluso cuando cambian repentinamente debido a su naturaleza no paramétrica, al contrario que otras basadas en NN.

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810

6 REFERENCIAS

[1]. Shahriari, Bobak, et al. "Taking the human out of the loop: A review of Bayesian optimization." *Proceedings of the IEEE* 104.1 (2015): 148-175.

[2]. Williams, Christopher KI, and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. No. 3. Cambridge, MA: MIT press, 2006.

[3]. Bega, Dario, et al. "CARES: Computation-aware scheduling in virtualized radio access networks." *IEEE Transactions on Wireless Communications* 17.12 (2018): 7993-8006.

[4]. Zhao, Nan, et al. "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks." *IEEE Transactions on Wireless Communications* 18.11 (2019): 5141-5152.

[5]. Ayala-Romero, Jose A., et al. "vrAIn: Deep learning based orchestration for computing and radio resources in vRANs." *IEEE Transactions on Mobile Computing* 21.7 (2020): 2652-2670.

[6]. O-RAN Alliance, "O-RAN-WG1-O-RAN Architecture Description- v01.00.00." Technical Specification, Februry 2020.

[7]. Raca, Darijo, et al. "On leveraging machine and deep learning for throughput prediction in cellular networks: Design, performance, and challenges." *IEEE Communications Magazine* 58.3 (2020): 11-17.

[8]. Gomez-Miguelez, Ismael, et al. "srsLTE: An open-source platform for LTE evolution and experimentation." *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. 2016.

[9]. Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971* (2015).

[10]. Li, Lihong, et al. "A contextual-bandit approach to personalized news article recommendation." *Proceedings of the 19th international conference on World wide web*. 2010.

[11]. Rusmevichientong, Paat, and John N. Tsitsiklis. "Linearly parameterized bandits." Mathematics of Operations Research 35.2 (2010): 395-411.

[12]. Duvenaud, David. *Automatic model construction with Gaussian processes*. Doctoral dissertation 2014.

[13]. Bull, Adam D. "Convergence rates of efficient global optimization algorithms." *Journal of Machine Learning Research* 12.10 (2011).

[14]. Krause, Andreas, and Cheng Ong. "Contextual gaussian process bandit optimization." *Advances in neural information processing systems* 24 (2011).

[15]. Srinivas, Niranjan, et al. "Gaussian process optimization in the bandit setting: No regret and experimental design." *arXiv preprint arXiv:0912.3995* (2009).

[16]. Berkenkamp, Felix, Andreas Krause, and Angela P. Schoellig. "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics." *Machine Learning* (2021): 1-35.

[17]. Fiducioso, Marcello, et al. "Safe contextual Bayesian optimization for sustainable room temperature PID control tuning." *arXiv preprint arXiv:1906.12086* (2019).

[18]. Sui, Yanan, et al. "Safe exploration for optimization with Gaussian processes." *International conference on machine learning*. PMLR, 2015.

[19]. Sui, Yanan, et al. "Stagewise safe bayesian optimization with gaussian processes." *International conference on machine learning*. PMLR, 2018.

[20]. Amani, Sanae, Mahnoosh Alizadeh, and Christos Thrampoulidis. "Regret bound for safe gaussian process bandit optimization." *Learning for Dynamics and Control*. PMLR, 2020.
[21]. Marquez, Cristina, et al. "Identifying common periodicities in mobile service demands with spectral analysis." *2020 Mediterranean Communication and Computer Networking Conference (MedComNet)*. IEEE, 2020.
[22]. Dulac-Arnold, Gabriel, et al. "Deep reinforcement learning in large discrete action spaces." *arXiv preprint arXiv:1512.07679* (2015).

Financiado por el Programa "Next Generation" de la Unión Europea en el marco del acuerdo n.º 2022/0004810