UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

21|22

Trabajo fin de grado

Aprendizaje por refuerzo mediante bandidos duelistas

Miguel González González

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Aprendizaje por refuerzo mediante bandidos duelistas

Autor: Miguel González González
Tutor: Pablo Castells Azpilicueta

mayo 2022

Miguel González González
Aprendizaje por refuerzo mediante bandidos duelistas

Miguel González González
C/ Franciso Tomás y Valiente nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Resumen

En el marco del aprendizaje por refuerzo, el problema del bandido multibrazo modela situaciones en las que un agente debe maximizar la recompensa obtenida como resultado de ejecutar acciones de un conjunto finito de posibilidades, cada una de las cuales produce valores distribuidos de manera distinta. Este problema permite modelar distintas situaciones prácticas en campos como la recomendación. No obstante, existen otras situaciones en las que es mucho más difícil obtener de manera fiable los valores numéricos de recompensa que determinar, dado un par de acciones, cuál de las dos ha generado una cantidad mayor. Motivado por este hecho, surge la variante de los **bandidos duelistas**, en las que el agente debe determinar qué acción es la mejor seleccionando parejas para comparar y obteniendo únicamente un valor binario que indica cual de las dos resulta vencedora. Esta variante no solo es de relevancia cuando es difícil extraer la recompensa numérica individual, sino también en situaciones donde no tiene sentido llevar a cabo acciones individuales y solo es posible efectuar comparaciones por parejas.

Con el fin de profundizar en el estudio de esta variante, se selecciona un amplio conjunto de políticas para el agente y se determina, mediante experimentación simulada en casos variados, cuáles de ellas desempeñan mejor y en qué situaciones, así como qué configuraciones de hiperparámetros de las mismas permiten alcanzar un mayor rendimiento y cómo influye el hecho de no tener acceso a la recompensa numérica frente a los agentes del problema del bandido multibrazo que si lo tienen. Para llevar a cabo los experimentos, se ha desarrollado un marco de simulación en el que se han implementado las políticas seleccionadas y los escenarios necesarios para dar respuesta a las preguntas planteadas.

Palabras clave

Bandidos duelistas, bandido multibrazo, aprendizaje por refuerzo, aprendizaje automático.

Abstract

Within the scope of reinforcement learning, the multi-armed bandit problem models situations in which an agent must maximize a reward value resulting from carrying out actions from a finite set of posibilities, each one producing values distributed differently. This problem models various practical situations in fields such as recommender systems. However, there exist other situations where it is harder to accurately measure the numerical reward values. Meanwhile, determining, among two actions, which one produced a better value, is way simpler. Due to this, a new version of the problem appears, named the **dueling bandits** variant. The agent must determine which action is the best one via selecting pairs of actions and obtaining as a consequence a binary value indicating which one won. This variant is not only of relevance when it is hard to extract a numerical individual reward value, but also in situations where it does not make sense in the first place to carry out individual actions, and only paired ones are possible.

In order to study this variant, we select a large set of different policies for the agent and we determine, via simulated experimentation in multiple situations, which ones perform better and on which situations, as well as which hyperparameter configurations allow for an optimal performance and how does the fact of not having access to the numerical reward values impact on the performance compared to standard multi-armed bandits. In order to carry out the experiments, we develop a simulation framework in which we implement the selected policies and the required environments to answer the posed questions.

Keywords

Dueling bandits, multi-armed bandit, reinforcement learning, machine learning.

Índice

1

Introducción

En este Trabajo de Fin de Grado se estudia el problema de los **bandidos duelistas**, variante del problema del bandido multibrazo dentro de la categoría del aprendizaje por refuerzo. Para ello, se realiza experimentación simulada con el fin de evaluar y optimizar el rendimiento de los principales algoritmos existentes en la literatura en una amplia variedad de situaciones. Dichos algoritmos a comparar son seleccionados a partir de una extensa revisión bibliográfica y recopilados en este trabajo. Con el fin de poder realizar los experimentos de modo flexible y cubrir todos los casos deseados, se diseña e implementa un marco de simulación para bandidos duelistas propio en el que se incluyen implementaciones de los algoritmos elegidos.

1.1. Motivación

Dentro del aprendizaje por refuerzo, el **problema del bandido multibrazo** modela situaciones en las que se debe maximizar la recompensa obtenida a través de llevar a cabo acciones escogidas de un conjunto finito predeterminado de ellas. Cada acción conlleva con su ejecución la obtención de una recompensa distinta, de modo que el agente que se enfrenta al problema debe encontrar el equilibrio entre la búsqueda de la mejor acción y la ejecución repetida de la misma para extraer la mayor cantidad de recompensa posible.

A través de este problema, es posible modelar diversas situaciones que se dan en la práctica relacionadas con sistemas informáticos. Por ejemplo, en sistemas de recomendación puede interpretarse una función de ránking como una acción a escoger, y la satisfacción de los usuarios como consecuencia de dicho ránking (medida, como es habitual, a través de métricas como *precisión* o *recall*) como la recompensa que reporta la elección. El objetivo de maximizar la recompensa se traduce en maximizar la satisfacción de los usuarios y la calidad de las recomendaciones efectuadas. Sin embargo, existen otras situaciones en las que medir directamente este valor de recompensa no es lo natural, pero determinar, dado un par de acciones, cuál desempeña mejor es mucho más sencillo. Un ejemplo es la realización de *tests A/B* para determinar, de entre un conjunto de sistemas, el más preferible, mediante comparaciones por parejas realizadas dividiendo a los usuarios en dos grupos aleatorios, cada uno de

los cuales accede a un sistema distinto, y contrastando los resultados obtenidos por cada uno de ellos.

Motivado por estas situaciones, surge una variante del problema conocida como **bandidos duelistas** [18]. En ella, el agente debe escoger parejas de acciones a realizar, obteniendo como recompensa únicamente un valor binario que representa cuál de los acciones resulta ganadora. Para esta variante se han desarrollado variadas políticas para el agente, siguiendo distintas aproximaciones, que deben enfrentarse a la problemática de distinguir la calidad de cada acción efectuando el menor número posible de comparaciones por pares.

Este problema presenta utilidad no solo para modelar situaciones en las que es más fácil efectuar comparaciones por pares, sino también en los casos en los que no existe manera alguna de ejecutar acciones individuales, solamente pudiendo llevarse a cabo por parejas. Un ejemplo de esto son torneos de juegos que se llevan a cabo por enfrentamientos entre pares de oponentes. Más concretamente, puede considerarse cada jugador del torneo como una de las acciones, y es evidente entonces que solo pueden realizarse comparaciones entre dos acciones, sin existir la posibilidad de llevarlas a cabo individualmente. En este contexto, el problema de los bandidos duelistas se traduce en encontrar al mejor jugador de manera fiable con el menor número de comparaciones. En estos casos, además, pueden ocurrir situaciones en las que no haya un ganador único a causa de que haya acciones que desempeñen muy bien contra algunas, pero muy mal contra otras, sin existir ninguna que sea superior globalmente.

Para remarcar la distinción que caracteriza a los bandidos duelistas, continuando con el ejemplo de los torneos de juegos, identificar qué corredor es superior en carreras contrarreloj es un problema que puede modelarse en el marco de los bandidos multibrazo (dado que se puede medir directamente un valor numérico que representa su calidad), mientras que determinar qué equipo de fútbol es el mejor en una liga corresponde a un problema de bandidos duelistas, al no ser posible extraer valores individuales sino únicamente comparaciones binarias como resultado de enfrentamientos por parejas.

Además de la motivación dada por estar presente en estas situaciones prácticas, esta variante presenta gran interés matemático como problema abstracto, consistente en identificar el mejor candidato de un conjunto que no puede observarse directamente ni muestrearse individualmente, sino que exclusivamente pueden obtenerse muestras por parejas acerca de la calidad comparativa de los elementos del mismo.

1.2. Objetivos

El objetivo del trabajo es llevar a cabo un estudio experimental de los distintos agentes presentes en la literatura en la mayor cantidad de situaciones posible, con el fin de dar respuesta a distintas preguntas sobre los mismos, y motivado por la falta de experimentos exhaustivos que abarquen la gran variedad de situaciones y políticas que han sido presentadas desde la introducción del problema.

Para ello, en este TFG se estudia en primer lugar la literatura de investigación del problema, seleccionando una gran variedad de algoritmos a comparar, cada uno de ellos con variados puntos de vista a la hora de ser diseñados. Dicha selección se realiza mediante una revisión bibliográfica, partiendo inicialmente de un listado de artículos proporcionados por el tutor del trabajo, posteriormente ampliada a partir de otras fuentes, obtenidas tanto mediante búsqueda de las palabras clave relevantes como a través de las referencias presentes en los artículos revisados, que posteriormente han sido filtradas para restringirse a revistas y congresos de primer nivel según las recomendaciones del tutor. Una vez realizada la selección, se plantea como objetivo analizar una amplia variedad de situaciones para determinar cuáles de las políticas escogidas desempeñan mejor y bajo qué condiciones, así como qué configuraciones de hiperparámetros permiten maximizar el rendimiento obtenido.

Para poder dar respuesta a las preguntas con la mayor flexibilidad posible, se plantea la realización de los experimentos de manera simulada. Por tanto, otro de los objetivos del trabajo, ante la falta de implementaciones lo suficientemente extensas, es la creación de un marco que permita llevar a cabo estas simulaciones, lo que conlleva asimismo la inclusión de todos los algoritmos a analizar en dicho marco.

1.3. Estructura de la memoria

La memoria se estructura como sigue. En el Capítulo 2, se define de manera precisa el problema y se presentan los principales algoritmos para los agentes seleccionados de la literatura. Para cada algoritmo, se explica su funcionamiento (incluyendo un pseudocódigo ilustrativo), las suposiciones adicionales que realiza y las garantías teóricas existentes sobre el mismo. Este capítulo se divide en dos secciones: en la Sección 2.1 se presenta el problema original del bandido multibrazo y los principales agentes, con el fin de proveer contexto y presentar las bases que se utilizan posteriormente. En la Sección 2.2 se analiza el problema de los bandidos duelistas, que el objeto de estudio principal de este trabajo. Cada una de estas secciones incluye como última subsección (Subsección 2.1.6 y Subsección 2.2.9) un breve resumen que incluye una tabla comparativa de las políticas presentadas. A continuación, el Capítulo 3 se centra en los experimentos llevados a cabo. Las primeras secciones (Sección 3.1 y Sección 3.2) están dedicadas a concretar los objetivos a llevar a cabo y la metodología seguida para ello. A continuación, el resto de las secciones se centran en presentar los resultados experimentales obtenidos para cada uno de los objetivos. Finalmente, en el Capítulo 4 se establecen las conclusiones extraídas y se presentan posibles futuras líneas de trabajo en base a estas. Se incluye un apéndice (Apéndice A) en el que se proporcionan detalles sobre el diseño y la implementación del marco de simulación utilizado para los experimentos y se enlaza al repositorio donde puede encontrarse el código del mismo.

2

Estado del arte

2.1. El problema del Bandido Multibrazo

El bandido multibrazo es un problema clásico dentro del **aprendizaje por refuerzo**, que es el estudio de problemas en los que un **agente** interactúa con un **entorno** realizando **acciones** escogidas de un conjunto determinado y que conllevan como resultado una señal de **recompensa** y un posible cambio en el entorno. En estos problemas, el agente debe *aprender* por sí solo cómo comportarse a partir de las interacciones que realiza.

En el caso del bandido multibrazo, al agente se le presentan n opciones, o *brazos* (en analogía con los de una máquina tragaperras), de entre los cuales debe elegir cuál accionar. Al hacerlo, recibe una recompensa numérica aleatoria cuya distribución depende únicamente del brazo accionado. El entorno no cambia a lo largo del tiempo, y el objetivo es maximizar la recompensa acumulada a lo largo de sucesivas tiradas.

Incluso en un entorno tan simple como este, se pone de manifiesto la problemática de la **exploración frente a explotación**. Si el agente desea encontrar el brazo que mejores recompensas proporciona, forzosamente deberá *explorar*, es decir, accionar distintos brazos hasta asegurarse de que uno de ellos es ventajoso. Al hacerlo, pospone la oportunidad de *explotar*, en otras palabras, accionar repetidas veces el brazo que considera más provechoso con el fin de maximizar la recompensa. Una de las consideraciones principales a tener en cuenta en relación al comportamiento de un agente es cómo equilibra estos dos aspectos.

Más allá del modelo teórico, este problema está presente en numerosos escenarios de la realidad. Por ejemplo, en el marco de los sistemas de recomendación, es posible considerar cada artículo a recomendar como una de las acciones de un bandido multibrazo, siempre y cuando sea posible medir una recompensa numérica derivada de dicha recomendación. El agente, entonces, buscará mostrar al usuario los artículos que produzcan una mejor reacción por parte del mismo. La *exploración*, en este marco, se traduce en presentar productos nuevos al usuario con el fin de producir un mayor interés que los que ya conocía.

El bandido multibrazo ha sido estudiado ampliamente en la literatura, por ejemplo, en el segundo capítulo de [14]. A continuación, se presentarán los aspectos más importantes del mismo, así como algunos de los principales algoritmos que puede seguir un agente para optimizar la recompensa.

2.1.1. Definición del problema

Un problema del bandido de N brazos está dado por N distribuciones aleatorias de valores reales, $(R_1, \ldots, R_N)$, llamadas **distribuciones de recompensa**, asociadas a cada uno de los N brazos. Cada una de las distribuciones tiene una propiedad característica, llamada **valor** del brazo, que consiste en la media del mismo: $\mu_j = \mathbb{E}[R_j]$.

El problema se desarrolla por rondas, comúnmente denominadas *épocas*. En la ronda T-ésima, el agente selecciona el brazo k-ésimo de acuerdo a un criterio predefinido (en el cual pueden tenerse en cuenta los resultados de las $T-1$ rondas que preceden a esta) y obtiene una **recompensa** r_T, muestreada de R_k. El objetivo del agente es obtener la mayor cantidad de recompensa posible a lo largo de su ejecución.

Para evaluar el comportamiento del agente, es posible considerar distintas métricas. Una de las principales en la literatura es el **regret**:

$$\rho(T) = \mu^* - r_T, \tag{2.1}$$

donde $\mu^* = \max\{\mu_1, \ldots, \mu_N\}$ es el mejor valor posible. El regret mide cómo de alejada está la recompensa de la ronda con respecto al valor medio óptimo. Así, si en una ronda dada T el agente escoge el mejor brazo, se tendrá que $\mathbb{E}[\rho(T)] = 0$. El objetivo es, por tanto, minimizar el regret. Una buena estrategia del agente debería garantizar que $\rho(T) \to 0$ con probabilidad 1 cuando $T \to \infty$.

Puesto que otro de los objetivos deseables es alcanzar el brazo óptimo lo más rápido posible, no basta con atender al valor del regret en un punto dado para determinar si el agente es bueno, sino también el coste invertido para llegar a dicho valor. Para solventar esto, puede definirse el **regret acumulado**:

$$\hat{\rho}(T) = \sum_{t=1}^{T} \rho(t) = T\mu^* - \sum_{t=1}^{T} r_t. \tag{2.2}$$

Al tratarse de la suma de todos los valores históricos del regret, comparar el valor de $\hat{\rho}$ para dos agentes en un mismo momento permite evaluar todo su desempeño y no solo el de la ronda en cuestión. El objetivo de optimización en cuanto a $\hat{\rho}$ es que crezca lo más lento posible.

Además del regret, es posible considerar otras métricas de desempeño, tanto en su variante puntual como acumulada. Las más intuitivas son el propio valor de recompensa: $r(T) = r_T$, que se busca

maximizar, o bien, si solo interesa encontrar el mejor de los brazos, el valor $\omega(T) = \delta_{k_T k^*}$, donde k_T es el índice (del 1 al N) del brazo elegido en la ronda T, k^* es el índice del brazo con valor μ^*, y δ_{ij} vale 1 si $i = j$ y 0 en otro caso. Denotaremos $\hat{r}$ y $\hat{\omega}$ a las versiones acumuladas de estas métricas.

Finalmente, aunque en muchos de los estudios teóricos no se considera, es importante prestar atención al **horizonte máximo** T_M, que es el número de rondas que un agente puede llevar a cabo. Es posible que $T_M = \infty$, pero en la práctica es deseable que el comportamiento sea bueno para intervalos de tiempo acotados.

2.1.2. Agentes básicos

En esta sección, como primer paso para atacar el problema, se presentarán brevemente ideas sencillas para el comportamiento del agente. Vamos a considerar que, tras la ronda T, el agente almacena el valor de la recompensa obtenida, y mantiene una estimación empírica del valor de cada brazo $\hat{\mu}_k(T) = \frac{\sum_{t \in P_k(T)} r_t}{N_k(T)}$, donde $P_k(T)$ es el conjunto de rondas en las que el brazo k fue accionado, y $N_k(T) = |P_k(T)|$ es el número de veces que lo ha sido. En otras palabras, el agente mantiene un vector de medias muestrales del valor de cada brazo según sus observaciones. Si $N_k(T) = 0$, puede tomarse un valor prefijado inicial para $\hat{\mu}_k(T)$, por ejemplo, 0.

Una primera aproximación *voraz* es escoger en cada ronda el mejor brazo obtenido hasta el momento, es decir, tirar del brazo con índice $\arg\max_k\{\hat{\mu}_k(T)\}$. Sin embargo, este comportamiento de *explotación pura* impide que el agente encuentre el mejor brazo: se quedará estancado indefinidamente en el primer brazo que obtenga un valor de $\hat{\mu}$ mayor que el valor inicial. La alternativa contraria, de *exploración extrema*, consistiría en elegir el brazo completamente al azar. Esto impediría obtener un buen valor de recompensa.

El método más sencillo que reúne estas dos estrategias es el conocido como ε**-voraz** o ε***-greedy***. Como hiperparámetro, se prefija un valor $\varepsilon \in [0, 1]$, que representa la probabilidad de exploración. En cada ronda, con probabilidad ε se elige explorar, tomando un brazo al azar, y con probabilidad $1 - \varepsilon$ se elige explotar, eligiendo el mejor brazo hasta el momento. Para una cantidad ilimitada de rondas, todos los brazos se acabarán probando un número suficiente de veces como para que se encuentre el mejor. A partir de ahí, se escogerá la mejor acción con probabilidad $1 - \varepsilon$. Así, cuanto mayor sea ε, más rápido se dará con la mejor opción, pero una vez encontrada seguirá habiendo una probabilidad ε de escoger una opción aleatoria, siendo la recompensa obtenida en este caso un valor subóptimo.

Esto presenta dos desventajas claras, derivadas de elegir entre exploración y explotación siempre de la misma manera. La primera es que se tarda demasiado en encontrar el mejor brazo, puesto que la exploración es aleatoria y no depende de la incertidumbre que se tiene sobre el brazo ni del valor estimado hasta el momento. La segunda es que, una vez encontrado el mejor brazo, no se explota lo suficiente puesto que se continúan realizando exploraciones.

Estos problemas se manifiestan en las métricas de rendimiento, como podemos analizar de forma sencilla en el regret acumulado. Supongamos que el algoritmo comenzase en el caso ideal en el que las estimaciones son perfectas, y por tanto $\hat{\mu}_k = \mu_k$. Entonces:

$$\mathbb{E}(\hat{\rho}(T)) = \mathbb{E}\left(T\mu^* - \sum_{t=1}^{T} r_t\right) = T\mu^* - \sum_{t=1}^{T} \mathbb{E}\left(r_T\right). \tag{2.3}$$

Como $\mathbb{E}(r_T) = \varepsilon \cdot \alpha + (1-\varepsilon)\mu^*$, donde $\alpha = \frac{\sum_{k=1}^{N} \mu_k}{N}$, dado que con probabilidad ε se explora cualquiera de los brazos al azar, sigue de (2.3) que $\mathbb{E}(\hat{\rho}(T)) = \varepsilon \cdot T \cdot (\mu^* - \alpha) = O(T)$. Es decir, el regret acumulado crece de forma lineal.

Una primera aproximación para resolver ambos problemas es asignar un valor muy pequeño de exploración, pero forzar al agente a realizar un número de pasadas iniciales por todos los brazos. De esta manera, el agente dedica la primera parte de su estrategia puramente a explorar, pudiendo permitirse una mayor explotación después. Esta idea, aunque sencilla, puede usarse como paso previo a cualquier otra estrategia que se desee llevar a cabo, y permite un cierto *guiado* inicial al agente. Otra posibilidad es permitir que el valor de ε varíe con el tiempo. Si se toma ε como una función decreciente del número de épocas transcurridas desde el inicio, con $\varepsilon \to 0$ si $T \to \infty$, se consigue reducir el efecto de la exploración cuando se tiene más certeza de que se ha encontrado el brazo óptimo.

Si bien esta discusión inicial sirve para ilustrar la problemática, el comportamiento de estas estrategias no es bueno dada su excesiva rigidez. El agente, para desempeñar adecuadamente, debe poseer una mayor riqueza en su algoritmia a la hora de elegir los brazos, evitando simplemente alternar entre exploración y explotación puras. A continuación se presentarán, para terminar con la discusión sobre bandidos multibrazo, dos extendidos algoritmos para el agente que proporcionan muy buenos resultados y cuyas ideas serán explotadas en el problema modificado de los bandidos duelistas.

2.1.3. Upper Confidence Bound (UCB)

El algoritmo UCB se basa en asignar una puntuación a cada brazo combinando el valor estimado del mismo con la incertidumbre que se tiene acerca de dicho valor. Tras ello, el agente escogerá el brazo con la mejor puntuación. Es decir, tras la ronda T, se calculan los valores de puntuación:

$$s_k(T) = \hat{\mu}_k(T) + v_k(T), \tag{2.4}$$

donde $v_k(T)$ debe ser una función adecuada que estime la incertidumbre de la aproximación $\hat{\mu}_k(T)$. Con esos valores, se escoge para la siguiente ronda el brazo $\arg\max_k\{s_k(T)\}$. Un valor típico para $v_k(T)$, aunque no el único posible, es:

$$v_k(T) = \gamma \sqrt{\frac{\log(T)}{N_k(T)}}, \tag{2.5}$$

donde γ es una constante. Por un lado, cuantas más veces se haya tirado del brazo (que es cuando más acertada será la estimación de $\hat{\mu}_k(T)$) mayor será el valor de $N_k(T)$ y por tanto, disminuirá $v_k(T)$. Por otro lado, el término $\log(T)$ permite, al aumentar con el tiempo, que cada brazo se pueda escoger un número no acotado de veces, asegurando que cuando $T \to \infty$, el brazo óptimo global sea encontrado. Finalmente, el valor γ es un hiperparámetro que permite regular el efecto de $v_k(T)$ sobre $s_k(T)$, de tal forma que se pueda favorecer o desfavorecer la exploración según se requiera.

Asimismo, como $v_k(T)$ es máximo si $N_k(T) = 0$, el algoritmo realiza de manera natural una primera pasada por todos los brazos, técnica que favorece un mejor arranque, como se explicó en la sección anterior. En el Algoritmo 2.1 se resume la estrategia.

Entrada: $N, T, \hat{\mu}_k(T-1), N_k(T-1), \gamma$.
Salida : Índice del brazo a accionar en la ronda T

```
for k ← 1 to N do
    if N_k(T−1) = 0 then
        return k
    end
    s_k ← μ̂_k(T−1) + γ · sqrt(log(T−1) / N_k(T−1))
end
return arg max_k(s_k)
```

Algoritmo 2.1: Algoritmo UCB para un agente que se enfrenta al bandido multibrazo.

Pese a la sencillez algorítmica de UCB, su flexibilidad aporta resultados mucho mejores que los algoritmos de la Subsección 2.1.2. En [2], se demuestra que $\mathbb{E}(\hat{\rho}(T)) = O(\log(T))$, es decir, el crecimiento del regret acumulado es logarítmico y por tanto mejor que los algoritmos ε-voraces.

2.1.4. Thompson Sampling

Otra aproximación, que es la llevada a cabo en la estrategia conocida como Thompson Sampling, es seguir un enfoque bayesiano, asumiendo que todas las R_K forman parte de una misma familia de distribuciones aleatorias que varían únicamente en su media μ_K. Es decir, $R_K = R(\mu_K)$ donde R es una distribución paramétrica única (por ejemplo, gaussiana o Bernoulli). Dicho párametro, la media, **se modela como una variable aleatoria**, dado que no se tiene conocimiento de su valor real. El algoritmo mantiene una estimación de la distribución de cada μ_K, que denotaremos $P_T(\mu_K)$ (distribución *a priori*), donde el subíndice indica que se trata de la estimación al comienzo de la ronda T. Ahora, si en esa ronda se obtiene información del brazo K-ésimo como consecuencia de haberlo accionado, se debe actualizar la distribución utilizando la metodología Bayesiana usual para el cálculo de la distribución *a posteriori*:

$$P_{T+1}(\mu_K) \propto P_T(\mu_K) \cdot R(r_T|\mu_K). \tag{2.6}$$

Cabe observar que el valor de la constante de proporcionalidad no es relevante, dado que el objetivo es comparar los brazos entre sí, y todos ellos tendrán la misma constante. El agente, para maximizar la recompensa inmediata, debería escoger el brazo cuyo valor esperado, $\mathbb{E}(P_{T+1}(\mu_K))$, sea máximo. Pero, para no perjudicar la exploración, la mejor alternativa es simplemente muestrear cada uno de los $P_{T+1}(\mu_K)$ y escoger el que mayor resultado proporcione. Así, se escogerán brazos de los que se tenga poca certeza, dado que la distribución será menos localizada y por tanto, al muestrear, se podrán obtener valores más grandes. Al mismo tiempo, una vez se haya explorado lo suficiente, al muestrear se obtendrá con gran probabilidad el brazo más alto.

A la hora de implementar el algoritmo en la práctica, es importante escoger una distribución $P_0(\mu_K)$ adecuada, que facilite el cálculo de las sucesivas distribuciones *a posteriori*. Por ejemplo, si R tiene una distribución Bernoulli, es decir, $\mathbb{P}[R_K = 0] = (1 - \mu_K)$ y $\mathbb{P}[R_K = 1] = \mu_K$, la teoría Bayesiana de las distribuciones conjugadas indica que lo mejor es tomar para $P_0(\mu_K)$ una distribución Beta, que está dada por la siguiente función de densidad:

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \tag{2.7}$$

donde α y β son constantes, y $B(\alpha, \beta)$ es el valor necesario para que la densidad integre 1. Esta distribución es la ideal porque, comenzando con $P_0(\mu_K)$ siendo una distribución Beta de parámetros α_0 y β_0, permite calcular $P_{T+1}(\mu_K)$ de manera inmediata, resultando ser de nuevo una distribución Beta con parámetros $\alpha_0 + S_K(T)$, $\beta_0 + F_K(T)$, donde $S_K(T)$ es el número de *éxitos*, o veces que el brazo K proporcionó recompensa 1, y $F_K(T)$ el número de *fracasos*, o veces que el brazo K proporcionó recompensa 0.

Con este análisis teórico, por tanto, el algoritmo de Thompson Sampling para brazos con distribución Bernoulli queda como se describe en el Algoritmo 2.2.

Entrada: N, T, α_0, β_0, resultados de las rondas previas.
Salida : Índice del brazo a accionar en la ronda T

1 **for** $k \leftarrow 1$ **to** N **do**
2 $S_k \leftarrow$ número de veces que el agente obtuvo 1 tirando de k hasta ahora.
3 $F_k \leftarrow$ número de veces que el agente obtuvo 0 tirando de k hasta ahora.
4 $\theta_K \leftarrow$ muestrear aleatoriamente de distribución Beta $(\alpha_0 + S_k, \beta_0 + F_k)$
5 **end**
6 **return** $\arg\max_k(\theta_k)$

Algoritmo 2.2: Algoritmo Thompson Sampling con distribución *a priori* Beta.

Al igual que en el caso de UCB, se ha demostrado en [7] que la tasa de crecimiento del regret acumulado es $\hat{\rho}(T) = O(\log(T))$, mejorando los algoritmos sencillos de la Subsección 2.1.2. Asimismo,

en [7], se realizan experimentos que muestran que Thompson Sampling obtiene mejores resultados de regret acumulado que UCB en las condiciones consideradas.

Existen varias alternativas para aplicar Thompson Sampling cuando las distribuciones de recompensa son continuas (a diferencia de la Bernoulli). Una de ellas, computacionalmente menos eficiente, es sustituir la distribución Beta por otra distribución adecuada (por ejemplo, si la recompensa sigue una distribución normal, la *a priori* también deberá ser normal aunque será necesario realizar más cálculos para obtener sus parámetros). Otra alternativa es mantener el Algoritmo 2.2, introduciendo un *umbral de éxito* que determine si cada tirada realizada contabiliza para S_k o F_k. En otras palabras, se discretiza la variable de recompensa, convirtiéndola en una distribución Bernoulli, para poder aplicar el algoritmo. Esto aporta una mayor eficiencia computacional aunque añade un nuevo hiperparámetro.

2.1.5. Otros algoritmos y variaciones

A lo largo de esta sección se ha presentado la variante principal del problema del bandido multibrazo, así como algunos de los algoritmos para su resolución. Aunque lo desarrollado aquí es una parte representativa del problema y será suficiente como fundamentos para el resto del trabajo, cabe destacar que este problema ha sido estudiado ampliamente en la literatura y existen multitud de algoritmos adicionales y variantes.

En cuanto a otros agentes, uno de los principales es **EXP3**, introducido en [4], que acciona los brazos con una distribución de probabilidad que comienza siendo uniforme pero se va modificando con pesos que dependen de manera exponencial del conocimiento adquirido sobre el brazo. Otros agentes, por mencionar algunos, son **SOFTMAX**, **POKER** y **LEAST-TAKEN** [15].

Por otra parte, existen variantes del problema que, manteniendo la estructura básica descrita en la Subsección 2.1.1, añaden elementos que aportan complejidad al modelo. Una de las más importantes es el **bandido multibrazo contextual** [11], en el que se presenta al agente un *vector de contexto* en cada época, y la recompensa de los brazos depende no solo del número de brazo sino también de dicho vector. El agente debe entonces estimar correctamente el valor de cada brazo en función del contexto. Otro ejemplo es el del **bandido adversario** [3], en el que el entorno también cambia en cada época, siendo el valor de cada brazo elegido por un *adversario* (el bandido multibrazo *usual* es un caso particular de este problema, en el que el *adversario* siempre elige la misma estructura para los brazos).

2.1.6. Resumen

El problema del bandido multibrazo modela situaciones en las que es necesario escoger una acción de entre varias disponibles, cada una de las cuales produce una recompensa aleatoria distribuida de manera distinta, inicialmente desconocida. El objetivo es maximizar la recompensa obtenida, para lo

que es necesario equilibrar un comportamiento exploratorio (para determinar cuál es la mejor opción sin estancarse en una subóptima) con la explotación de las recompensas de la acción que ha sido estimada como la mejor.

De este modo, una buena estrategia para un agente que busca resolver el problema comenzará generalmente con una exploración de las distintas acciones y, conforme el tiempo avanza y la certeza de las estimaciones aumenta, tornará hacia la explotación de la opción estimada como la mejor. A lo largo del capítulo se ha profundizado en las principales estrategias para ello, resumidas en la Tabla 2.1.

Algoritmo	**Estrategia resumida**	**Regret**
ε-greedy	Escoger aleatoriamente entre explorar (elegir una acción cualquiera) o explotar (elegir la mejor acción hasta el momento)	$O(T)$
UCB	Utilizar intervalos de confianza para equilibrar exploración y explotación: el extremo derecho de un intervalo de confianza es mayor tanto si la estimación es mayor como si lo es la incertidumbre (varianza).	$O(\log(T))$
Thompson Sampling	Enfoque bayesiano: aprender una distribución para el valor de cada acción, que converge al valor real conforme se obtienen muestras de la recompensa.	$O(\log(T))$

Tabla 2.1: Resumen de algoritmos para bandidos multibrazo.

2.2. Bandidos duelistas (Dueling Bandits)

El problema de los bandidos duelistas fue introducido en el año 2009 por Y. Yue y T. Joachims [18]. Se trata de una modificación del problema clásico del bandido multibrazo presentado en el capítulo previo, en el que no es posible obtener un valor numérico preciso para la recompensa obtenida como resultado de accionar uno de los N brazos. En su lugar, solo es posible realizar enfrentamientos entre parejas de brazos, accionando ambos y obteniendo como consecuencia un resultado binario que indica cuál de los dos vence la comparación.

De esta manera, se busca modelar de una mejor manera los problemas de aprendizaje por refuerzo en línea que surgen en sistemas de recuperación de información y motores de búsqueda, entre otros. Una de las motivaciones para ello es la existencia de metodologías para obtener, a partir de experimentos interactivos con usuarios, realimentación acerca de cuál de dos *rankings* de resultados en un sistema de recuperación de la información es mejor, siendo mucho más difícil asignar valores cardinales a un *ranking* individual según la respuesta del usuario. Un ejemplo de estas metodologías se presenta en [10].

En general, el problema de los bandidos duelistas sirve para modelar situaciones en las que, o bien existe un valor de recompensa *subyacente* a tomar una decisión (accionar un brazo), pero no

es posible observar dicho valor, o bien no existe tal valor subyacente, y solo tiene sentido enfrentar decisiones entre ellas, siendo imposible ejecutar cada una de manera aislada. En este segundo caso pueden ocurrir situaciones más complejas, como por ejemplo que no exista una acción óptima o que existan ciclos de preferencias (es decir, que la acción 1 venza frente a la 2, que a su vez gana a una acción 3 pero la acción 3 sea superior a la 1), que requerirán especial estudio y consideración a la hora de diseñar estrategias para el agente.

Un ejemplo de problema que puede ser modelado mediante bandidos duelistas es determinar qué jugador de tenis es mejor a partir de un conjunto dado, en el menor número de enfrentamientos posibles. En este ejemplo, la única manera de medir la calidad de un jugador es mediante enfrentamientos por parejas, de modo que los bandidos multibrazo no pueden aplicarse (es imposible extraer un valor numérico de un tenista de manera aislada) y debe recurrirse a los bandidos duelistas.

En esta sección se presenta, en primer lugar, el problema básico de los bandidos duelistas, introduciendo las definiciones, elementos, métricas y notaciones relevantes. A continuación, se describen distintos algoritmos para el agente que buscan resolver el problema, estudiando sus premisas y posibles limitaciones. Finalmente, se explica cómo pueden influir las circunstancias especiales mencionadas en el párrafo anterior, fruto de no existir un valor de recompensa subyacente, y se presentan algoritmos especialmente diseñados para estos casos.

2.2.1. Definición del problema

El objetivo del problema es encontrar al mejor de N brazos o acciones. A diferencia del problema de bandidos multibrazo descrito en la Subsección 2.1.1, en lugar de tener N distribuciones aleatorias numéricas para cada brazo, se dispone de N^2 distribuciones Bernoulli por cada pareja (i, j), $1 \leq i, j \leq N$, cuyo parámetro es la **probabilidad de que** i **venza a** j, denotada por P_{ij}. Se define asimismo la **ventaja de** i **frente a** j como $\varepsilon_{ij} = P_{ij} - \frac{1}{2} \in [-\frac{1}{2}, \frac{1}{2}]$. Entonces, es necesario que para todo par de brazos i, j, se tenga:

$$\varepsilon_{ij} = -\varepsilon_{ji}, \tag{2.8}$$

lo que además implica que $\varepsilon_{ii} = 0$. De esta manera se garantiza que el orden en el que se enfrentan los brazos no influye en el resultado de la confrontación.

Existen restricciones adicionales en el problema que facilitan el diseño de los agentes que lo resuelven, pero pueden no darse en la situación real que se esté modelando. Un aspecto a considerar en los algoritmos que se utilicen será entonces cuáles de estas restricciones, que definimos a continuación, asumen para su correcto funcionamiento.

En primer lugar, se tiene la **existencia de ganador Condorcet**, es decir, un brazo i^* tal que $\varepsilon_{i^*j} \geq 0$

para todo j. En caso de que exista, es el objetivo a encontrar por el agente. Si no, será necesario definir otras nociones de *ganador* como objetivo se desea buscar. Esta última situación se desarrolla en la Subsección 2.2.7, aunque para el resto del trabajo se asumirá que existe ganador Condorcet salvo que se indique lo contrario.

Aún más restrictiva es la **suposición de orden total** en los brazos, es decir, que exista una ordenación *de mayor a menor* que garantice $\varepsilon_{ij} \geq 0$ toda vez que i sea mayor que j en la ordenación. En este caso, existe un ganador Condorcet que es simplemente el máximo de la ordenación. Además, pueden exigirse dos restricciones adicionales. Por un lado, la **transitividad estocástica**, que requiere que si i es mejor que j y éste mejor que k, entonces:

$$\varepsilon_{ik} \geq \max\{\varepsilon_{ij}, \varepsilon_{jk}\}. \tag{2.9}$$

Es decir, impone que los valores de probabilidad del modelo sean *consistentes* en el sentido de que no es posible que un brazo i gane más veces a un brazo j que las que gana a un tercero k que es peor que el j. Por otro lado, la **desigualdad triangular estocástica**, que requiere que si el brazo i es mejor que el j y este mejor que el k, entonces:

$$\varepsilon_{ik} \leq \varepsilon_{ij} + \varepsilon_{jk}. \tag{2.10}$$

Esta última condición es notablemente restrictiva, y solo se cumple en ciertos modelos concretos para las distribuciones por parejas. Un ejemplo de tal modelo es asumir que las cantidades P_{ij} proceden de un bandido multibrazo clásico subyacente, invisible para el agente, de tal manera que si, como en la Subsección 2.1.1, se tienen variables aleatorias $\{R_1, \ldots, R_N\}$ para las recompensas de cada acción, entonces $P_{ij} = \mathbb{P}[R_i > R_j]$. En otras palabras, se asume que las comparaciones se realizan ejecutando ambas acciones y comparando los valores de recompensa obtenidos entre sí, aunque cabe recalcar que **estos valores no son mostrados en ningún momento al agente**. Si, además, las distribuciones de recompensa son normales (y por tanto, entre otras propiedades, la probabilidad de empatar una comparación es 0), se cumple la Ecuación 2.8, de modo que los valores de P_{ij} resultantes son válidos para el problema de bandidos duelistas, y, ordenando los brazos de mayor a menor por su valor de $\mu_i = \mathbb{E}[R_i]$, se verifican también todas las suposiciones adicionales mencionadas anteriormente.

Una vez definido el modelo, el objetivo del agente, al igual que en el problema clásico, es encontrar la mejor de las acciones, es decir, el ganador Condorcet descrito anteriormente. Para ello, en la ronda T, el agente puede seleccionar una pareja (i_T, j_T) de brazos, y obtiene un valor binario indicando si el brazo i_T venció al j_T. Dicho valor sigue una distribución Bernoulli con probabilidad $P_{i_T j_T}$. A partir de dichas comparaciones, el agente debe tratar de identificar cuál es la mejor acción y ejecutar enfrentamientos que la incluyan. En otras palabras, la **exploración** consiste en comparar parejas desconocidas con el fin de determinar qué brazo es mejor, y la **explotación** consiste en comparar parejas de bra-

zos que se consideran buenos. La explicación para esto último es que las comparaciones, en ciertas aplicaciones reales, conllevan mostrar al usuario dos opciones (por ejemplo artículos o ránkings en un sistema de recuperación de información), y por tanto es deseable que sean buenas. Para evaluar el rendimiento del agente según estas consideraciones, si i^* es el ganador Condorcet, se define el **regret Condorcet**:

$$\rho_c(T) = \frac{1}{2}\left(\varepsilon_{i^* i_T} + \varepsilon_{i^* j_T}\right), \tag{2.11}$$

que también puede considerarse en su versión acumulada $\hat{\rho}_c(T) = \sum_{t=1}^{T} \rho_c(t)$. Véase que esta cantidad lo que penaliza es escoger acciones (i_T, j_T) que son peores en comparación con el óptimo i^*. Por otro lado, se consigue minimizar el regret si se encuentra el brazo óptimo y se escoge a partir de ahí la pareja (i^*, i^*).

En ciertas situaciones prácticas puede no ser esencial que ambas elecciones de brazo sean óptimas. Por ejemplo, en el ámbito de la recuperación de información, quizás baste con que una de las dos opciones presentadas al usuario sea buena para garantizar una correcta experiencia. En estos casos es conveniente utilizar otra métrica de rendimiento, conocida como **regret Condorcet débil**:

$$\rho_{wc}(T) = \min\{\varepsilon_{i^* i_T}, \varepsilon_{i^* j_T}\}, \tag{2.12}$$

que solo considera la mejor de las dos opciones escogidas, evitando penalizar al agente cuando solamente una de las dos acciones es mala. Bajo esta métrica es posible llevar a cabo otras estrategias como, por ejemplo, utilizar uno de los dos brazos de la comparación para seguir explorando mientras que en el otro brazo se presenta la mejor opción hasta el momento. La función de regret a utilizar dependerá del problema que se desee modelar.

En las secciones que siguen, se analizarán diversas estrategias para el agente, así como las suposiciones que realizan y los resultados teóricos que garantizan. Estas son las que, en el siguiente capítulo, serán puestas en práctica a través de experimentos para comprobar su rendimiento empírico en distintas situaciones.

2.2.2. Interleaved Filter (IF)

El primer algoritmo para resolver el problema de los bandidos duelistas se conoce como **Interleaved Filter** y fue descrito por primera vez por Yue & Joachims [17]. El algoritmo actúa en dos fases: en primer lugar, se realizan las suficientes comparaciones entre brazos como para determinar con suficiente confianza cuál es el ganador estimado $\hat{i}$. Una vez seleccionado, el algoritmo solamente realiza enfrentamientos $(\hat{i}, \hat{i})$, con el fin de minimizar el regret.

Una tal estructura, que podría describirse como *explorar y luego explotar*, requiere que el paso

1 sea lo suficientemente robusto como para evitar entrar en el paso 2 con una solución subóptima. En el caso de *Interleaved Filter*, la política seguida conlleva excluir los brazos de manera sucesiva hasta quedarse con el mejor de ellos. Cabe destacar que este proceso de eliminación es definitivo, es decir, cuando el algoritmo determina que uno de los brazos no es el óptimo, lo descarta de manera permanente y no se vuelve a accionar en épocas posteriores. Para realizar los descartes, la estrategia es mantener en todo momento un brazo **candidato** a óptimo, y enfrentarlo con los demás brazos, para, una vez adquirida suficiente confianza en las estimaciones del rendimiento del candidato, mantenerlo o eliminarlo garantizando con alta probabilidad que tras cada ronda el candidato no empeore. Asimismo, los brazos que tienen alta probabilidad de ser peores que el candidato actual son eliminados.

Una de las ventajas de esta aproximación es que en cada enfrentamiento, uno de los dos brazos seleccionados es el que el algoritmo considera mejor hasta el momento (el candidato), de tal modo que en el paso de la exploración el regret no se ve tan perjudicado como si se hicieran enfrentamientos aleatorizados. Asimismo, dadas la transitividad y desigualdad triangular estocásticas (cf. Ecuación 2.9, 2.10), es posible ahorrar gran cantidad de comparaciones, dado que los brazos que pierden con un candidato pueden ser retirados cuando este es eliminado, independientemente de la confianza del algoritmo en la estimación de rendimiento de dichos brazos. Esto es así porque para eliminar al candidato es necesario que haya otro brazo mejor con suficiente confianza, y la transitividad estocástica garantiza entonces que los brazos que pierden con el candidato según las estimaciones tengan alta probabilidad de ser peores (independientemente de la confianza de las mismas). Esta técnica se añade al algoritmo y se conoce como **poda**.

Por otro lado, esta política presenta ciertas desventajas. En primer lugar, se asume la existencia de un ganador único Condorcet, así como transitividad y desigualdad triangular estocásticas (cf. Subsección 2.2.1) para el correcto funcionamiento. Otro de los principales problemas es que no existe recuperación ante los posibles errores en la estrategia: si la acción óptima es eliminada del proceso de manera incorrecta, no existe ningún modo de que vuelva a ser elegida más adelante. De hecho, la política de *explorar y luego explotar* ocasiona que, a partir de un instante de tiempo, se elija únicamente un brazo, de tal manera que si este es subóptimo, el regret quedará estancado para siempre en un valor no nulo.

Para solventar esto último, es necesario realizar las suficientes comparaciones como para garantizar que el brazo óptimo se mantiene con muy alta probabilidad. Esto ocasiona que, aunque los resultados teóricos de regret sean buenos, en la práctica tarden muchas iteraciones en alcanzarse, como se mostrará en los experimentos del capítulo siguiente. El algoritmo busca equilibrar este problema mediante la introducción de un hiperparámetro de horizonte máximo, T_M, que permite decidir si explorar durante un gran número de rondas garantizando la alta probabilidad de encontrar el óptimo, o bien acelerar el proceso de eliminación con el fin de acabar antes de un número dado de iteraciones.

Para estimar la probabilidad de victoria del brazo i frente al j tras cada comparación, se calcula el

valor $\hat{P}_{ij}$ como el número de victorias de i entre el número total de enfrentamientos. Como medidor de confianza, se utiliza una técnica similar a la de UCB (cf. Subsección 2.1.3), en la que se calcula el valor:

$$v_{ij}(T) = \sqrt{\frac{\log(T_M \cdot N^2)}{N_{ij}(T)}}, \tag{2.13}$$

donde T_M es el horizonte máximo, N el número de brazos y $N_{ij}(T)$ el número de comparaciones entre i y j realizadas hasta la ronda T. Después, este valor se utiliza de tal manera que si $\hat{P}_{ij} + v_{ij}(T) < 1/2$, se determina que i pierde contra j. Se observa de la expresión de $v(T)$ que cuantas más comparaciones se hayan hecho hasta el momento, más pequeño será su valor (representando una mayor confianza en la estimación). El valor del numerador permite, como se anticipó en el párrafo anterior, disminuir el impacto de v_{ij} si se dispone de pocas rondas para explorar (T_M bajo), o si hay pocos brazos (N bajo), al haber menos comparaciones posibles.

Los detalles de la estrategia, que integran toda la discusión realizada, pueden comprobarse en el Algoritmo 2.3. El principal resultado teórico de [17] garantiza que el regret acumulado está acotado por una función $O(\frac{N}{\varepsilon_{i^* i^{**}}} \log(T))$, donde i^{**} es el segundo mejor brazo. De este resultado cabe observar que cuanto mayor sea la distinción entre el brazo óptimo y el siguiente mejor (y, por transitividad estocástica, entre todos los demás), menor sera el regret incurrido, como es razonable. Además, la probabilidad de que el algoritmo no devuelva el brazo óptimo, para un horizonte $T_M >> N$, es de $\frac{1}{T_M}$.

2.2.3. Beat The Mean (BTM)

Una de las características esenciales del algoritmo *Interleaved Filter* discutido en la sección previa es su dependencia de la transitividad estocástica en las distribuciones de los brazos. Yue & Joachims [19] proponen un nuevo algoritmo, llamado **Beat The Mean** (BTM), que no solo permite relajar la necesidad de dicha transitividad, sino que también proporciona mejores resultados en general. Esto es de gran importancia dado que se ha mostrado que una distribución de preferencias que verifique la Ecuación 2.9 de la Subsección 2.2.1 rara vez se da en la práctica.

En esta estrategia se mantiene la mecánica eliminatoria de expulsar a los brazos que pierden con alta probabilidad frente a uno dado (la forma de estimar el rendimiento de los mismos, así como la confianza en dicha estimación, es idéntica a la de *Interleaved Filter* descrita en la Subsección 2.2.2). Para poder desprenderse de la transitividad estocástica, se evita que estos enfrentamientos sean contra el *mejor brazo* y, en su lugar, se hacen frente a un *brazo medio* ficticio, que representa una acción cuya recompensa es, en cierto modo, el promedio de entre todas las que hay en juego.

Por lo tanto, se requieren dos modificaciones frente a *Interleaved Filter*. La primera es que el brazo seleccionado para ser explorado (que, para mayor eficiencia en la exploración, será el que menos com-

Entrada: T_M (horizonte máximo), N (número de brazos)
Salida : $\hat{\imath}$ (mejor brazo estimado)

$i \leftarrow 1$ // Candidato
$W \leftarrow \{2, \ldots, N\}$ // Brazos no eliminados
$\hat{P}_{ij} \leftarrow \frac{1}{2}$ // Estimaciones de probabilidades
$N_{ij} \leftarrow 0$ // Número de comparaciones entre cada par
while $W \neq \emptyset$ **do**
  // Ronda para obtener información de los brazos
  for $j \in W$ **do**
    $N_{ij} \leftarrow N_{ij} + 1$
    Enfrentar (i, j) y actualizar $\hat{P}_{ij} = \frac{\text{Victorias de i contra j}}{N_{ij}}$
    $v_j \leftarrow \sqrt{\frac{\log(T_M \cdot N^2)}{N_{ij}}}$ // Valor de confianza
    if $\hat{P}_{ij} - v_j > \frac{1}{2}$ **then** $W \leftarrow W \setminus \{j\}$;
  end
  // Determinar si alguno de los brazos supera al candidato para sustituirlo
  for $j \in W$ **do**
    if $\hat{P}_{ij} + v_j < \frac{1}{2}$ **then**
      //Poda de perdedores frente a i, independientemente de la confianza:
      for $k \in W \setminus \{j\}$ **do**
        if $\hat{P}_{ik} > \frac{1}{2}$ **then** $W \leftarrow W \setminus \{k\}$;
      end
      $W \leftarrow W \setminus \{i\}$; $i \leftarrow j$; // Sustitución del candidato
      Resetear todos los $\hat{P}_{rs}$ y N_{rs}
    end
  end
end
return i

Algoritmo 2.3: Algoritmo Interleaved Filter.

paraciones haya realizado hasta el momento) es enfrentado contra otro brazo de los restantes escogido al azar. Este oponente aleatorizado representa ese *brazo medio*, dado que en cada emparejamiento se escoge uno distinto de manera equiprobable. Una vez realizada la comparación, se actualizan las estimaciones $\hat{P}_k$ del brazo que se estaba explorando dependiendo de si gana el enfrentamiento. Cabe destacar que, en esta ocasión, basta con mantener una estimación por brazo (en lugar de una por pareja de brazos), que representa la calidad frente a la media.

Tras cada comparación, si el valor de $\hat{P}$ del peor de los brazos es mucho menor que el del mejor de ellos, en el sentido de que difieren en mayor medida que un cierto margen de confianza $v(T)$, se elimina permanentemente al peor de ellos, y todos los enfrentamientos que hubiera realizado son descontados retroactivamente de las estimaciones $\hat{P}$, de tal manera que sigan representando el rendimiento frente a la media de los brazos restantes. La estrategia completa, que incluye algunos detalles como cambios en el cálculo del intervalo de confianza $v(T)$, se consigna en el Algoritmo 2.4.

Puesto que ya no se compara en cada ronda con el mejor de los brazos, el regret podría verse perjudicado en las fases iniciales de la exploración (dado que uno de los brazos empleados es aleatorio),

Entrada: T_M (horizonte máximo), N (número de brazos), γ (relajación de transitividad)
Salida : $\hat{i}$ (mejor brazo estimado)

$W \leftarrow \{1, \ldots, N\}$ // Brazos no eliminados
$\hat{P}_i \leftarrow \frac{1}{2}$ // Estimaciones de probabilidades
$N_i \leftarrow 0$ // Número de comparaciones entre cada par
while $|W| > 1$ **do**
  // Explorar el brazo con menos candidatos, enfrentándo con otro al azar.
  $i \leftarrow \arg\min_{k \in W}\{N_k\}$; $j \leftarrow$ valor aleatorio de W ; $N_i \leftarrow N_i + 1$
  Enfrentar (i, j) y actualizar $\hat{P}_i = \frac{\text{Victorias de i}}{N_i}$
  // Determinar peor y mejor brazo
  $i \leftarrow \arg\min_{k \in W}\{\hat{P}_k\}$; $j \leftarrow \arg\max_{k \in W}\{\hat{P}_k\}$
  // Determinar valor de intervalo de confianza
  $n \leftarrow \min_{k \in W}\{\hat{N}_k\}$
  $v \leftarrow 3 \cdot \gamma^2 \cdot \sqrt{\frac{1}{n}\log(2T_M \cdot N)}$
  if $\hat{P}_i + v < \hat{P}_j - v$ **then**
    $W \leftarrow W \setminus \{i\}$ // Eliminar perdedor
    Eliminar enfrentamientos frente a i de todos los $\hat{P}_k$ y N_k.
  end
end
return $W[0]$

Algoritmo 2.4: Algoritmo Beat The Mean.

pero a cambio esto permite prescindir de la transitividad estocástica. En concreto, basta con que se cumpla, en lugar de la Ecuación 2.9 de la Subsección 2.2.1, la siguiente modificación:

$$\gamma \cdot \varepsilon_{ik} \geq \max\{\varepsilon_{ij}, \varepsilon_{jk}\}, \tag{2.14}$$

donde γ es un parámetro de relajación que puede ser superior a 1. El regret acumulado está acotado por una función $O(\frac{\gamma^7 \cdot N}{\varepsilon_*}\log(T))$, donde ε_* es la menor de las ventajas entre el mejor brazo y el resto. Nótese que la cota empeora según el grado γ de relajación en la transitividad. La probabilidad de que el algoritmo no devuelva el brazo óptimo sigue siendo $\frac{1}{T_M}$.

2.2.4. Algoritmos basados en bandidos multibrazo

Las estrategias consideradas en las secciones anteriores han sido diseñadas *desde cero* teniendo en cuenta las características del problema de los bandidos duelistas, mediante un agente que es capaz de observar los resultados de los emparejamientos y elegir los dos brazos siguientes en consecuencia. En esta sección se explora la idea alternativa de reducir el problema de bandidos duelistas al tradicional de bandido multibrazo, pudiendo así utilizar toda la teoría y métodos existentes en esa área. Esta idea fue presentada por primera vez en [1], y radica en el concepto de **combinar múltiples agentes de bandido multibrazo**, de tal manera que en su conjunto puedan resolver el problema de los enfrentamientos por parejas. Por lo tanto, un hiperparámetro inherente a este sistema es qué estrategia

multibrazo se utiliza en los agentes subyacentes.

Una primera aproximación heurística es considerar que cada una de las dos acciones que se comparan es elegida por un agente multibrazo de manera independiente. Es decir, se tienen dos agentes multibrazo, A_1 y A_2, cuyo objetivo es elegir la acción que venza al brazo seleccionado por su contrincante. Cada agente realiza su elección y recibe como recompensa numérica un valor 0 si pierde frente a la elección del rival, o un valor 1 si gana. Es decir, las distribuciones de recompensa de cada brazo para cada uno de los agentes es Bernoulli. Cabe destacar que el agente no conoce en ningún momento qué brazo está eligiendo el rival. De esta manera, cada uno de los agentes, de manera individual, trata de encontrar el mejor de los N brazos, haciendo que las comparaciones realizadas sean cada vez de mejor calidad. Esta estrategia, mostrada en el Algoritmo 2.5, se conoce como **sparring** [1] y, aunque no tiene resultados teóricos que garanticen su buen desempeño, en la práctica obtiene resultados extremadamente buenos, como se mostrará en el capítulo siguiente.

Entrada: N (número de brazos), M_1 (agente multibrazo), M_2 (agente multibrazo)

```
//Inicializar dos agentes multibrazo de N brazos
A_1 ← M_1(N); A_2 ← M_2(N)
while True do
    i ← acción(A_1); j ← acción(A_2)
    Enfrentar (i, j)
    Recompensar A_1 con 0 si i pierde, con 1 en caso contrario. Igual con A_2 y j.
end
```

Algoritmo 2.5: Algoritmo Sparring.

La segunda alternativa es mantener N agentes multibrazo, $\{A_1, \ldots, A_N\}$, de tal manera que el objetivo de A_i sea escoger el mejor oponente para el brazo i. Con esta estrategia, se elige el primero de los dos brazos a enfrentar y se hace que el agente correspondiente seleccione cuál debe ser su oponente. Después, se recompensa al agente con un 1 si su elección fue correcta, es decir, si esta logró ganar. De esta manera, los agentes se ocupan de elegir rivales cada vez más buenos, elevando la calidad de los emparejamientos realizados. Asimismo, se toma como brazo a enfrentar el oponente elegido por un agente multibrazo en la ronda anterior, lo que contribuye a mejorar ambos lados del enfrentamiento. Esta estrategia, denominada **MultiSBM**, se presenta en el Algoritmo 2.6.

El último algoritmo de reducción a bandidos multibrazo, denominado **Doubler**, mantiene un único agente multibrazo A. Este agente se ocupa de seleccionar la mejor acción para uno de los dos lados del enfrentamiento, y el otro lado se escoge al azar con el fin de obtener la mayor información posible. Para ello, el conjunto de brazos del cual se puede escoger al azar, W, que comienza con un único brazo, se actualiza periódicamente con los brazos escogidos por A desde la última actualización. De esta manera, este segundo oponente forma parte de los brazos que más ha accionado A recientemente, de modo que se trata de los mejores candidatos hasta el momento. Estas actualizaciones se realizan tras $1, 2, 4, 8, \ldots$ rondas desde la anterior actualización, es decir, en potencias de dos cada vez mayores.

Entrada: N (número de brazos), M (agente multibrazo)
//Inicializar N agentes multibrazo de N brazos
$A_i \leftarrow M(N)$ para cada $i = 1, \ldots, N$.
// Brazo inicial
$j \leftarrow 1$
while *True* **do**
    $i \leftarrow j$ //Asignar al primer brazo el segundo brazo de la ronda previa.
    $j \leftarrow$ acción(A_i) //Obtener el contrincante.
    Enfrentar (i, j) ; Recompensar A_i con 1 si i pierde, con 0 en caso contrario.
end

Algoritmo 2.6: Algoritmo MultiSBM.

La política completa se muestra en el Algoritmo 2.7.

Entrada: N (número de brazos), M (agente multibrazo)
$A \leftarrow M(N)$. //Agente multibrazo.
$W = \{1\}$ //Conjunto inicial de oponentes.
$i \leftarrow 1$ //Indice de la potencia de 2 actual.
while *True* **do**
    reset(A)
    $W' \leftarrow \emptyset$
    for $j = 1, \ldots, 2^i$ **do**
        $i \leftarrow$ elegir equiprobablemente de W ; $j \leftarrow$ acción(A)
        Enfrentar (i, j)
        Recompensar A con 1 si i pierde, con 0 en caso contrario.
        $W' = W' \cup \{j\}$
    end
    $W \leftarrow W'$; $i \leftarrow i + 1$
end

Algoritmo 2.7: Algoritmo Doubler.

2.2.5. Relative Upper Confidence Bound (RUCB)

Hasta ahora se han presentado, por una parte, algoritmos diseñados *desde cero* siguiendo la estrategia de *explorar y luego explotar*, que dedican un tiempo finito a encontrar el brazo óptimo y luego únicamente accionan ese brazo, y, por otra, algoritmos basados en bandidos multibrazo que reducen el problema a otros varios con recompensa numérica y equilibran exploración y explotación en la medida en la que el agente multibrazo subyacente lo haga. En esta subsección y en la siguiente, se presentan dos algoritmos que combinan lo mejor de los dos paradigmas: están diseñados *desde cero* para bandidos duelistas, teniendo así una mayor flexibilidad, pero se basan en las ideas presentes en los algoritmos para bandidos multibrazo, como, por ejemplo, el mantener intervalos de confianza para las probabilidades (del mismo modo que UCB, explicado en la Subsección 2.1.3), o utilizar distribuciones

a priori para los parámetros P_{ij} (al estilo de Thompson Sampling, visto en la Subsección 2.1.4), lo que permite equilibrar exploración y explotación a lo largo de todas las épocas, sin necesidad de detener la exploración en un instante prefijado. Concretamente, en esta subsección, se describe el algoritmo *Relative Upper Confidence Bound (RUCB)*, que es una potente adaptación de UCB al marco de los bandidos duelistas, presentada por primera vez en [21].

El algoritmo RUCB se basa, al igual que UCB, en asignar puntuaciones a cada enfrentamiento (i, j), combinando la estimación que se tiene sobre el resultado de i frente a j con un valor de confianza que disminuye con el número de veces que se ha realizado el enfrentamiento:

$$s_{ij}(T) = \hat{P}_{ij}(T) + v_{ij}(T), \tag{2.15}$$

donde $\hat{P}_{ij}(T)$ es la estimación empírica de la probabilidad de que i gane a j, calculada como es habitual, y $v_{ij}(T) = \sqrt{\alpha \frac{\log(T)}{N_{ij}(T)}}$ es el valor de confianza que crece con el tiempo (factor $\log(T)$), decrece con el número de veces que se ha efectuado el enfrentamiento $(N_{ij}(T))$ y cuyo impacto puede modificarse ajustando el factor de exploración α.

Una vez calculadas las puntuaciones, se trata de determinar si hay algún brazo que deba forzosamente ser elegido para el siguiente enfrentamiento. Un claro candidato sería un brazo i que cumpliera $s_{ij}(T) \geq \frac{1}{2}$ para todo rival j, dado que un valor superior a $\frac{1}{2}$ indica que, o bien el brazo tiene un rendimiento muy bueno, o bien se tiene mucha incertidumbre sobre su comportamiento. Así, el primer paso que realiza el algoritmo es construir el conjunto $\mathcal{C}$ de tales brazos. Si el conjunto $\mathcal{C}$ está vacío, no hay ningún candidato y el primer brazo a enfrentar se escoge al azar. Por otro lado, si el conjunto $\mathcal{C}$ consta de un solo brazo, se escoge ese para el emparejamiento, y además se recuerda para futuras rondas como **mejor brazo tentativo**.

En los casos en los que $\mathcal{C}$ tenga varios candidatos (por ejemplo, se sabe con certeza que un brazo rinde muy bien pero otro tiene aún mucha incertidumbre), se escoge el competidor al azar de entre los elementos de $\mathcal{C}$, con la salvedad de que se le asigna un peso mayor al *mejor brazo tentativo* de la ronda anterior, si lo hubiera. Una vez elegido el primer brazo a enfrentar, el segundo se escoge simplemente como aquel que tiene la mejor puntuación $s_{ij}(T)$ frente al que ya ha sido elegido. Estos detalles se resumen en el Algoritmo 2.8.

Como se introdujo al comienzo, RUCB es el primer algoritmo que puede ejecutarse equilibrando exploración y explotación sin límite de tiempo y sin utilizar agentes multibrazo subyacentes. Si se desea encontrar un brazo óptimo en tiempo finito para seguir el modelo *explorar y luego explotar*, basta con detener el algoritmo en un instante y quedarse con el brazo $\mathcal{B}$ (según la notación del Algoritmo 2.8) que se haya encontrado en ese momento, o el brazo i si no lo hubiese. Además, el regret acumulado es $O(\log T)$, como es deseable, asumiendo únicamente la existencia de ganador Condorcet (no se requieren consideraciones de transitividad dado que no se utilizan en la lógica del algoritmo).

Entrada: N (número de brazos), α (nivel de exploración)
$\mathcal{B} \leftarrow \emptyset$ // Hipotético mejor brazo
$\hat{P}_{ij} \leftarrow \frac{1}{2}$ // Estimaciones de probabilidades
$N_{ij} \leftarrow 0$ // Número de comparaciones entre cada par
for $T = 1, 2, \ldots$ **do**
  for $i, j \in [1, \ldots, N]^2$ **do**
    $s_{ij} \leftarrow \hat{P}_{ij} + \sqrt{\alpha \frac{\log(T)}{N_{ij}(T)}}$ // Cálculo de puntuaciones
  end
  $\mathcal{C} \leftarrow \{i \in [1, \ldots, N] : \forall j \in [1, \ldots, N], s_{ij} \geq \frac{1}{2}\}$
  if $|\mathcal{C}| = 0$ **then**
    $i \leftarrow$ muestrear equiprobablemente en $[1, \ldots, N]$
  else if $|\mathcal{C}| = 1$ **then**
    $i \leftarrow C[0]$; $\mathcal{B} \leftarrow i$ // Almacenamos el hipotético mejor brazo.
  else $i \leftarrow$ muestrear aleatoriamente en $\mathcal{C}$, dando a $\mathcal{B}$ probabilidad $\frac{1}{2}$ y el resto equiprobables. ;
  $j \leftarrow \arg\max_k \{s_{ki}\}$
  Enfrentar (i, j) y actualizar $\hat{P}_{ij}$; $N_{ij} \leftarrow N_{ij} + 1$
end

Algoritmo 2.8: Algoritmo RUCB.

2.2.6. Double Thompson Sampling (DTS)

El otro algoritmo basado en adaptar las ideas clásicas al problema de los bandidos duelistas es **Double Thompson Sampling (DTS)**, llamado así por ser la adaptación de Thompson Sampling (cf. Subsección 2.1.4) al escenario de los enfrentamientos por parejas, presentada en [16]. Como se discutió en la Subsección 2.1.4, la estrategia bayesiana de Thompson Sampling es particularmente eficiente cuando la variable de recompensa a modelar es binaria (con una distribución Bernoulli), de tal manera que este paradigma puede adaptarse con gran naturalidad a las comparaciones entre pares de brazos que se dan en el problema de los bandidos duelistas.

Siguiendo la línea de razonamiento del algoritmo original presentada en la Subsección 2.1.4, se asume que las probabilidades de victoria por parejas, P_{ij}, son variables aleatorias que siguen una distribución desconocida, modelada como una Beta (Ecuación 2.7) de parámetros $\alpha = 1 + S_{ij}$, $\beta = 1 + F_{ij} = 1 + S_{ji}$, donde S_{ij} es el número de *éxitos*, es decir, veces que i venció a j en el pasado, y F_{ij} el número de *fracasos*, o derrotas de i frente a j. Nótese que esto corresponde a tomar una distribución inicial (*a priori*) de parámetros $\alpha_0 = \beta_0 = 1$.

Para elegir el primer brazo a enfrentar, se muestrean valores para todos los P_{ij} de acuerdo a sus distribuciones estimadas según el párrafo anterior y se escoge el brazo que haya obtenido más valores por encima de $1/2$, es decir, se toma $\arg\max_i \sum_j \chi(P_{ij} > \frac{1}{2})$, donde $\chi(c)$ vale 1 si c se cumple y 0 si no. Para un mejor rendimiento, previo a este paso se realiza una *poda* del espacio de candidatos mediante puntuaciones RUCB (análogas a la Ecuación 2.15). Para ello, se calculan las puntuaciones s_{ij} y, a continuación, se calcula el conjunto de candidatos $\mathcal{C}$ como aquel que contiene a los brazos i

con mayor cantidad de $s_{ij} > \frac{1}{2}$. Este conjunto es el único que participa a la hora de seleccionar el primer brazo, lo que permite, por un lado, evitar tener que muestrear demasiadas variables aleatorias P_{ij} y, por otro, caer en comparaciones subóptimas a causa la naturaleza aleatoria del muestreo de P_{ij}.

Tras elegir el primer brazo, i, el segundo brazo se escoge teniendo en cuenta aquel cuyo valor P_{ji} muestreado es mejor. Los valores necesarios se vuelven a muestrear, en lugar de reciclar los usados para elegir el primer brazo. De nuevo, se realiza una poda previa mediante puntuaciones RUCB, de tal manera que solo se consideren para el segundo brazo aquellos con incertidumbre alta. En caso de no haber tales brazos, simplemente se toma de nuevo la acción i para el segundo brazo, dado que con alta probabilidad se tratará del óptimo. El procedimiento final se detalla en el Algoritmo 2.9. El algoritmo DTS es, en principio, uno de los mejores que existen en la actualidad, con una cota de regret $O(\log T)$ y excelentes resultados experimentales según sus autores [16]. En los experimentos del capítulo siguiente comprobaremos la eficiencia del algoritmo y analizaremos en qué casos funciona mejor. Otra de las grandes ventajas de DTS es que en ningún momento se asume la existencia de ganador Condorcet (puesto que siempre se escoge el brazo mediante una *votación* de los P_{ij}, sin requerir que todos sean mayores que $\frac{1}{2}$ como en RUCB), lo que debería ponerse de manifiesto en los resultados experimentales en situaciones sin tal ganador.

Entrada: N (número de brazos), γ (nivel de exploración)
$S_{ij} \leftarrow 0$ // Victorias de i frente a j
$N_{ij} \leftarrow 0$ // Número de comparaciones entre cada par
for $T = 1, 2, \ldots$ **do**
  // Cálculo de puntuaciones RUCB
  for $i, j \in [1, \ldots, N]^2$ **do**
    $s_{ij} \leftarrow \hat{P}_{ij} + \sqrt{\gamma \frac{\log(T)}{N_{ij}(T)}}$; $s'_{ij} \leftarrow \hat{P}_{ij} - \sqrt{\gamma \frac{\log(T)}{N_{ij}(T)}}$
    $P_{ij} \leftarrow$ muestrear de Beta$(1 + S_{ij}, 1 + S_{ji})$
  end
  $\hat{s}_i = \sum_{j \neq i} \chi(s_{ij} > \frac{1}{2})$
  $\mathcal{C} \leftarrow \{j \in [1, \ldots, N] : \hat{s}_j = \text{máx}_k\, \hat{s}_k\}$
  $i \leftarrow \arg\max_{k \in \mathcal{C}} \sum_{j \neq k} \chi(P_{kj} > \frac{1}{2})$
  for $j \in [1, \ldots, N]$ **do**
    if $j \neq i$ **then**
      $P_{ji} \leftarrow$ muestrear de Beta$(1 + S_{ji}, 1 + S_{ij})$
    else $P_{ii} \leftarrow \frac{1}{2}$;
  end
  $\mathcal{C}' \leftarrow \{j \in [1, \ldots, N] : s'_{ji} \leq \frac{1}{2}\}$
  $j \leftarrow \arg\max_{k \in \mathcal{C}'} P_{ki}$
  $N_{ij} \leftarrow N_{ij} + 1$
  Enfrentar (i, j) y actualizar S_{ij}, S_{ji} según el resultado.
end

Algoritmo 2.9: Algoritmo DTS.

2.2.7. Situaciones sin ganador Condorcet: algoritmo CCB

Hasta ahora, hemos asumido que existe una acción que supera a todas las demás, conocida como *ganador Condorcet* (aunque algunos algoritmos no la hayan requerido). Como se explicó en la Subsección 2.2.1, en el caso en el que exista una recompensa numérica subyacente a los brazos (aunque sea oculta al agente), se tiene un ganador Condorcet, correspondiente al brazo con mejor valor cardinal. No obstante, en la práctica, pueden aparecer distribuciones de preferencia en las que no existe una acción mejor que las demás, pudiendo existir ciclos (A es preferido sobre B, que es mejor sobre C, pero C supera a A). Un ejemplo de distribución sin ganador Condorcet es la mostrada en la Tabla 2.2.

P_{ij}	A	B	C	D
A	0.5	0.7	0.6	0.4
B	0.3	0.5	0.3	0.6
C	0.4	0.7	0.5	0.9
D	0.6	0.4	0.1	0.5

Tabla 2.2: Ejemplo de distribución por parejas sin ganador Condorcet, con $N = 4$ brazos. La posición en la fila i y columna j muestra el valor de P_{ij}, es decir, la probabilidad de que i gane a j. Al no haber ninguna fila con todos sus valores por encima de $0{,}5$, no existe ganador Condorcet.

La existencia de tales distribuciones presenta dos problemas principales. El primero es la necesidad de determinar qué brazo es el que debe encontrar el agente y definir una métrica que evalúe el grado de éxito en dicha búsqueda. El segundo es la posibilidad de que los algoritmos diseñados teniendo en mente la existencia de un brazo óptimo puedan fallar. Para resolver el primero, se define el **ganador Copeland** como aquel brazo que gana a la mayor cantidad de rivales, es decir, la acción con índice $\tilde{i} = \arg\max_i\{\sum_{j\neq i}\chi(P_{ij} \geq \frac{1}{2})\}$, que siempre existe (aunque puede no ser única). En cuanto a la métrica, si se calculase el regret usual tomando como brazo óptimo el ganador Copeland, se obtendrían posiblemente valores de regret negativos (dado que existen brazos que pierden con $\tilde{i}$), lo cual es indeseable. Para ello, una alternativa es la siguiente: en primer lugar, se asigna una **puntuación Copeland** a cada brazo en función del número de rivales a los que gana:

$$c_i = \frac{1}{N-1}\sum_{j\neq i}\chi\left(P_{ij} \geq \frac{1}{2}\right), \tag{2.16}$$

de tal manera que $c_i \leq c_{\tilde{i}}$ si $i \neq \tilde{i}$. Ahora, es posible definir como métrica el **regret Copeland**:

$$\rho_{nc}(T) = c_{\tilde{i}} - \frac{c_{i_T} + c_{j_T}}{2}, \tag{2.17}$$

donde (i_T, j_T) es el par elegido en el instante T. Así, la elección óptima es enfrentar dos ganadores Copeland. Se tiene, como es habitual, la versión acumulada de esta métrica: $\hat{\rho}_{nc}(T) = \sum_{t=1}^{T}\rho_{nc}(t)$.

De entre los algoritmos discutidos en este capítulo, distinguimos aquellos que utilizan como premisa

del diseño la existencia de ganador Condorcet:

- Los algoritmos *IF*, *BTM* y *RUCB* asumen la existencia de ganador Condorcet en su diseño. *IF* y *BTM* lo hacen en su naturaleza eliminatoria y su necesidad de transitividad. RUCB utiliza en su desarrollo un conjunto de candidatos calculado a partir de los potenciales ganadores Condorcet.
- Los algoritmos *Doubler*, *Sparring*, *MultiSBM* y *DTS* no asumen la existencia de ganador Condorcet en su diseño, aunque el único que presenta garantías teóricas para este caso [16] es **DTS**. Por tanto, consideraremos que este es el único algoritmo que prescinde de esta suposición completamente.

Existen algunas políticas para el agente diseñadas desde cero asumiendo que no existe ganador Condorcet, como el algoritmo *Copeland Confidence Bound (CCB)* [20]. Este es similar a RUCB dado que mantiene las puntuaciones s_{ij} de la Ecuación 2.15, así como las estimaciones inferiores s'_{ij} del Algoritmo 2.9, pero son utilizadas para calcular estimaciones de las puntuaciones Copeland c_i con las que determinar los emparejamientos. El primero de los brazos se escoge de entre los que mejores puntuaciones Copeland estimadas tienen, y el segundo se toma como aquel que aporta más información frente al primero. Para garantizar evitar estancarse en valores subóptimos (lo que es más probable al no haber ganador absoluto), ciertas decisiones se toman aleatoriamente. Además, se mantienen conjuntos por cada brazo i, que contienen los oponentes que aportan más información sobre el brazo i si se enfrentan a este. Todos estos detalles pueden consultarse en la Sección 4.1 de [20].

En cuanto a garantías de rendimiento, si bien el regret acumulado sigue siendo $O(\log T)$, cuando se introduce como variable el número de brazos N, la cota se convierte en $O(N^2 + N \cdot \log T)$, lo que conlleva una especial penalización cuando el número de brazos es alto, dado que el término cuadrático domina al término $N \log T$ para los valores de T razonables en la práctica.

2.2.8. Variaciones: *Top-k*, multiduelos y bandidos duelistas adversarios

El problema de los bandidos duelistas estudiado en esta sección se centra en todo momento en la identificación de la mejor acción (definida como el ganador Condorcet o el ganador Copeland). No obstante, existen otras variantes del problema ajustadas a distintas situaciones en las que el objetivo no es únicamente este. Una de ellas es la variante ***Top-***k [6], en la que el objetivo es identificar de manera eficiente el subconjunto de los $k \geq 1$ mejores brazos (ordenados por un criterio predefinido, como por ejemplo la puntuación Copeland (Ecuación 2.16)). Un método para lograrlo es partir de los algoritmos que mantienen una matriz de estimaciones de la distribución por parejas (como Interleaved Filter, Beat The Mean o RUCB) y extraer de dicha matriz los k mejores brazos en lugar de solamente el ganador. No obstante, dado que estos algoritmos buscan minimizar el regret, cuando tienen certeza de haber encontrado el ganador no invierten esfuerzo en distinguir el ranking *Top-*k, de modo que el resultado no es fiable. Para ello, una alternativa es modificar el procedimiento de muestreo y actualización de la matriz, de modo que garantice certeza en las estimaciones de los k mejores (a pesar de que ello conlleve un aumento en el regret). Esta idea da lugar al algoritmo **Preference-based Racing (PBR)** [6].

Otra de las variantes es la denominada **bandidos multiduelistas**. Del mismo modo que una de las motivaciones de los bandidos duelistas es la evaluación de funciones de *ranking* en sistemas de recuperación de información mediante el entrelazamiento de los resultados de parejas de tales funciones, existen métodos [5] de entrelazamiento de resultados de un número $m \geq 2$ de tales funciones para obtener, con menos datos, información de cómo esas m funciones se comparan entre sí. Motivado por ello, se introduce el problema de los bandidos multiduelistas [8], en el que el agente selecciona un subconjunto de m acciones en cada paso y obtiene como resultado todas las comparaciones por parejas de dicho subconjunto. Finalmente, al igual que en el caso del bandido multibrazo (cf. Subsección 2.1.5), es posible modelar matrices de preferencias cambiantes en el tiempo, es decir, entornos no estacionarios, a través del problema de los **bandidos duelistas adversarios** [13].

2.2.9. Resumen

El problema de los bandidos duelistas modifica el problema del bandido multibrazo impidiendo la observación del valor de recompensa por parte del agente. En su lugar, es posible accionar parejas de brazos y conocer cuál de los dos ha obtenido una mayor recompensa. Esta modificación está motivada por el hecho de que, en la práctica, es mucho más sencillo conocer las preferencias de un usuario de manera comparativa (es decir, saber qué artículo prefiere de entre dos opciones), que asignar un valor cardinal al nivel de preferencia de un artículo individual. El objetivo del problema es encontrar lo antes posible la mejor acción (aquella que vence a un mayor número de rivales). Por ello, de nuevo, debe equilibrarse la exploración (que ahora ha de hacerse por parejas) con la explotación del mejor brazo.

Naturalmente, el proceso de exploración por parejas puede resultar más o menos sencillo dependiendo de la estructura de las distribuciones de victoria por parejas. Por ello, se contemplan distintas suposiciones adicionales sobre las mismas. La **transitividad estocástica** (Ecuación 2.9) supone que si X vence mayoritariamente a Y e Y a Z, entonces X vence mayoritariamente a Z. La **desigualdad triangular estocástica** (Ecuación 2.10) restringe aún más la probabilidad de que X venza a Z, impidiendo que supere a la suma de las probabilidades de que X venza a Y e Y a Z. Finalmente, la **existencia de ganador Condorcet** asume que existe un brazo que vence a todos los demás.

Las estrategias discutidas en este capítulo pueden agruparse en tres categorías según el paradigma que siguen para atacar el problema:

1.– **Estrategias *ad-hoc*.** Diseñadas teniendo en cuenta desde un primer momento la estructura por parejas del problema y aprovechando las restricciones adicionales para minimizar el número de comparaciones.

2.– **Estrategias basadas en bandidos multibrazo.** Buscan reducir el problema a uno o varios problemas de bandido multibrazo para aplicar las estrategias ya conocidas y estudiadas.

3.– **Adaptaciones de bandidos multibrazo.** Toman las ideas de un algoritmo diseñado para el problema del bandido multibrazo y las adaptan a la situación de distribuciones por parejas.

Dichas estrategias se resumen en la Tabla 2.3.

Algoritmo	Categoría	Estrategia resumida	Suposiciones	Penalización
Interleaved Filtering	*Ad-hoc*	Mantener un *rey de la pista*. Los brazos que ganan lo reemplazan, los que pierden se eliminan.	Transitividad, triangular, Condorcet.	$O(\log(T))$
Beat the Mean	*Ad-hoc*	Similar a Interleaved Filtering, pero el *rey de la pista* no es el mejor brazo sino *la media* de los brazos no eliminados.	Transitividad (relajada), triangular, Condorcet.	$O(\log(T))$
Sparring	Basado en MAB	Asignar un agente multibrazo a cada una de las dos elecciones de contendientes y recompensarlo según su elección gana o pierde.	Condorcet.	Desconocido.
MultiSBM	Basado en MAB	Mantener un agente multibrazo por cada brazo, que debe escoger el mejor oponente para este. Recompensarlo si el oponente elegido vence.	Condorcet.	Depende del MAB subyacente.
Doubler	Basado en MAB	Mantener un agente multibrazo que escoja la mejor opción para uno de los lados del enfrentamiento. La otra se escoge aleatoriamente.	Condorcet.	Depende del MAB subyacente.
Relative UCB	Adaptación MAB	Adaptación de UCB a bandidos duelistas que mantiene intervalos de confianza para cada probabilidad de victoria por pares.	Condorcet.	$O(\log(T))$
Double Thompson Sampling	Adaptación MAB	Adaptación de Thompson Sampling a bandidos duelistas que mantiene distribuciones *a priori* para cada probabilidad de victoria por pares.	Ninguna.	$O(\log(T))$
Copeland Confidence Bound	Adaptación MAB	Adaptación de UCB a bandidos duelistas utilizando puntuaciones Copeland, que no asumen la existencia de un ganador absoluto (Condorcet).	Ninguna.	$O(\log(T))$

Tabla 2.3: Resumen de algoritmos para bandidos duelistas.

3

Experimentación y resultados

En este capítulo se presentan los experimentos desarrollados para evaluar la eficacia de los algoritmos descritos de manera teórica en el capítulo previo, así como los principales resultados extraídos. Estos experimentos están motivados por la falta de análisis prácticos exhaustivos en la literatura del problema dado que, pese a que con la introducción de nuevos algoritmos suelen realizarse pruebas experimentales para garantizar su eficiencia, el ámbito de dichas pruebas suele ser reducido, mostrando únicamente resultados de simulación en casos muy concretos que pueden no reflejar su desempeño general en otras situaciones, no pudiendo encontrarse ninguna comparativa completa con agentes y entornos variados. El principal fin con que se lleva a cabo esta parte experimental es dar respuesta a distintas preguntas relacionadas con la eficacia de los algoritmos presentados en situaciones variadas.

Los experimentos se han llevado a cabo en entornos simulados, esto es, las recompensas que genera cada brazo se obtienen por medio de muestreo aleatorio y no a partir de datos reales con usuarios. Esto permite que las pruebas se adapten precisamente a lo que se busca comprobar, sin que una posible falta de datos o inadecuación de los mismos impida la experimentación.

Para ello, ante la falta de librerías lo suficientemente flexibles que implementen algoritmos para el problema de los bandidos duelistas, se ha desarrollado un *framework* de simulación que permite llevar a cabo las pruebas que se ajustan a las preguntas que se buscan responder, además de incluir implementaciones de las políticas presentadas en el capítulo anterior y ser fácilmente ampliable para añadir posibles futuros experimentos que puedan surgir con futuros avances en el problema de los bandidos duelistas (por ejemplo, nuevas políticas para los agentes o entornos distintos).

3.1. Objetivos

Los experimentos se han diseñado y estructurado para atender tres objetivos principales. En primer lugar, **identificar qué política(s) desempeña(n) mejor** según la situación a la que se enfrenten. Para ello, se evaluará el *regret* acumulado de cada uno de los agentes llevando a cabo experimentos variando las siguientes características del problema:

1.– Número de brazos, incluyendo cantidades en rangos bajos y altos.

2.– Distribución de recompensas, variando entre la gaussiana (al ser un buen modelo de un problema continuo), la Bernoulli (utilizada en problemas binarios que son frecuentes en la práctica) y una distribución *ruidosa* (que afecte a las buenas propiedades estructurales que tienen las otras dos, con el fin de detectar qué algoritmos se ven perjudicados por ello).

3.– Modalidad de *regret* utilizado, comprobando si cambian los resultados cuando solo se toma el *regret* del mejor brazo seleccionado, como se discutió al final de la Subsección 2.2.1.

Una vez identificadas las políticas que destacan sobre las demás en cada situación, el segundo objetivo del análisis es la **configuración de hiperparámetros** que optimiza los resultados de tales políticas. De esta manera, se busca maximizar el rendimiento final obtenido. Para ello, se seleccionan las mejores políticas y se realizan experimentos variando las elecciones de hiperparámetros con el fin de evaluar las diferencias entre configuraciones.

Finalmente, se busca responder la pregunta de **cómo afecta la falta del conocimiento de las recompensas numéricas por parte de los agentes** definitoria del problema de los bandidos duelistas. Para ello, se enfrentan los mejores algoritmos resultantes de los experimentos anteriores contra algoritmos para el problema del bandido multibrazo, que tendrán acceso a las recompensas numéricas de los brazos, y se evalúan las diferencias entre ellos.

3.2. Metodología

Con el fin de ejecutar los experimentos que se ajusten a las preguntas planteadas en la sección de objetivos, se ha desarrollado una librería de simulación de bandidos duelistas que permite la definición de agentes y entornos así como la evaluación de su rendimiento a través de distintas métricas. Pueden encontrarse más detalles sobre la implementación, estructura y código de la misma en el Apéndice A.

Cada experimento consta de un conjunto $\mathcal{A}$ de *agentes* (políticas) que se enfrentan a un *entorno* (que define las distribuciones de los brazos), $\mathcal{E}$, durante un número de épocas prefijado T. Dada la naturaleza generalmente aleatoria de los agentes y los entornos, el experimento se repite un número K de veces, promediando los resultados de cada ejecución. A lo largo de cada repetición, se almacenan los valores de distintas métricas de evaluación, con el fin de generar las gráficas finales. Este proceso se resume en el pseudocódigo del Algoritmo 3.1.

La principal métrica utilizada durante los experimentos es una modificación [12] del regret Copeland acumulado, que asigna regret nulo a los ganadores Copeland y la máxima de las desventajas frente a los ganadores Copeland en caso contrario, es decir, el máximo de los valores $P_{ij} - \frac{1}{2}$, donde i es un ganador Copeland y j es el brazo seleccionado (se promedia este valor para ambos brazos de la pareja). Esta modificación es más adecuada que la presentada inicialmente con fines teóricos en la Subsección 2.2.7, dado que coincide con el regret Condorcet habitual en las situaciones en las que exista ganador Condorcet. De esta manera, con esta métrica es posible comparar cualquier política en cualquier entorno de manera precisa.

```
Entrada: 𝒜, ℰ, T, K
Salida : Gráficas con las métricas deseadas.
for n ∈ {1, ..., K} do
    for a ∈ 𝒜 do
        for t ∈ {1, ..., T} do
            (i, j) ← step(a)
            result ← pull(ℰ, i, j)
            reward(a, i, j, result)
            Actualizar métricas de a con i, j, result.
        end
        reset(ℰ); reset(a)
    end
end
```

Algoritmo 3.1: Bucle principal de un experimento.

Salvo en las situaciones en las que se indique lo contrario, los experimentos se llevan a cabo con N brazos gaussianos de varianza $\sigma^2 = 1$ y medias equidistantes $1, 2, \ldots, N$.

Aunque el entorno $\mathcal{E}$ de un experimento sea fijo, cuando sea necesario llevar a cabo comparaciones en distintos entornos (por ejemplo, para comprobar el efecto del número de brazos en el rendimiento de los agentes), se ejecutan varios experimentos y se agregan las métricas obtenidas en cada uno de ellos.

3.3. Comparativa de políticas

En esta sección se compara el rendimiento de las siguientes políticas para el agente:

- *Interleaved Filtering* y *Beat The Mean* con parámetro de horizonte igual al número de épocas del experimento.
- *Doubler*, *MultiSBM* y *Sparring* con agente multibrazo subyacente *Thompson Sampling* con $\alpha_0 = \beta_0 = 1$. Se escoge *Thompson Sampling* frente a *UCB* dado que trabaja con recompensas binarias (dadas por el resultado del brazo i frente al j) a las que *Thompson Sampling* se adecúa.
- *DTS* con $\gamma = 1$.
- *RUCB* y *CCB* con $\alpha = 0{,}51$, dado que en [21] se indica que cualquier valor de α superior a $1/2$ es válido y la cota proporcionada es mejor para valores bajos.
- Agente aleatorio que sirve como línea base.

Realizamos un primer experimento con el entorno fijado a $N = 20$ brazos para comprobar cómo resulta la comparativa. El experimento se lleva a cabo durante $T = 10000$ épocas para observar el comportamiento de los agentes tanto a corto como a largo plazo. Las curvas de regret resultantes se muestran en la Figura 3.1. Observamos el crecimiento lineal del agente aleatorio, mientras que el resto de curvas muestran, como se esperaba del análisis teórico, un crecimiento logarítmico, a excepción de la curva de *Doubler* que a esta escala de tiempo se comporta de manera lineal. Se observa que existen

grandes diferencias en el regret acumulado a largo plazo por parte de los agentes, siendo *sparring* el mejor de ellos con gran diferencia, seguido por *DTS*. A corto plazo (menos de 1000 épocas) *Doubler* es una opción competitiva junto a *DTS* y *sparring*, pero su posición empeora en regímenes de tiempo superiores. Cabe observar también que *Beat The Mean* requiere un gran número de épocas para mejorar al aleatorio. Aunque *CCB* está diseñado para mejorar a *RUCB* en el caso no transitivo, incluso en entornos transitivos como este se observa una ligera mejora alrededor de las 4000 épocas. Cabe remarcar que el mejor resultado lo obtiene una de las políticas más simples, *sparring*, sin garantía teórica demostrada en la literatura.

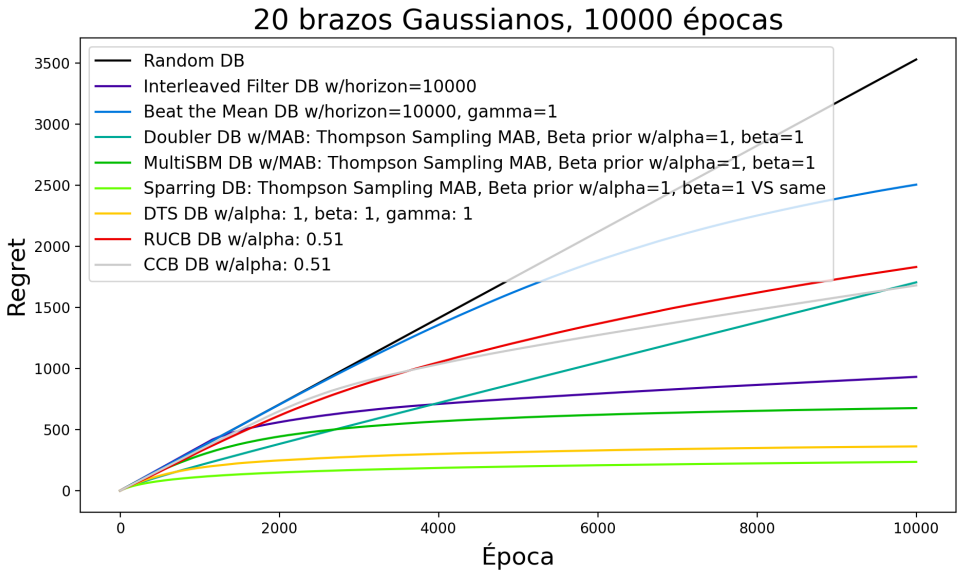

Figura 3.1: Regret acumulado en un entorno de $N = 20$ brazos gaussianos y $T = 10000$ épocas.

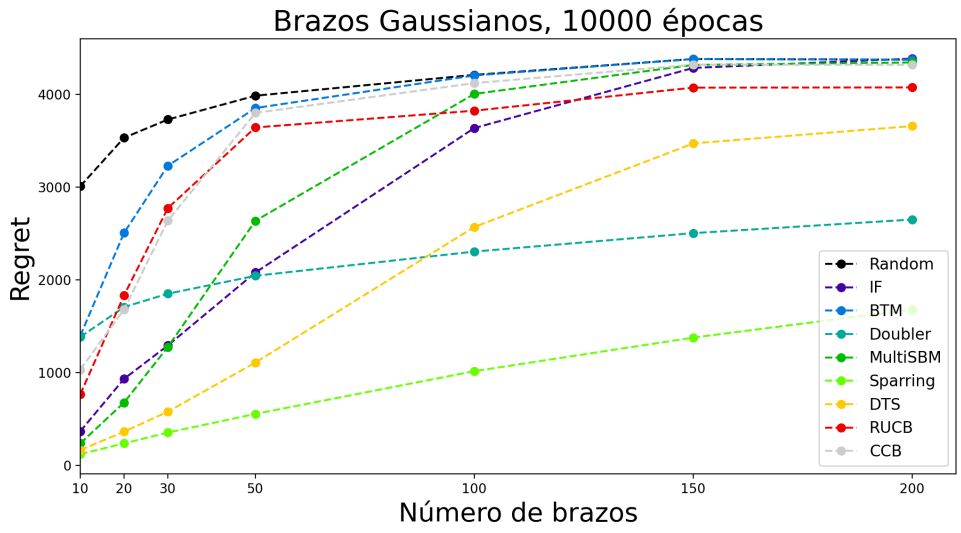

Figura 3.2: Regret acumulado para distinto número de brazos gaussianos y $T = 10000$ épocas.

A continuación buscamos comparar el efecto del número de brazos en el rendimiento. Para ello, repetimos el experimento anterior variando este hiperparámetro del entorno y representamos el regret final de cada agente en la Figura 3.2. De nuevo, el mejor de los agentes resulta ser *sparring* con gran diferencia. Cabe observar la robustez de *Doubler* frente al incremento del número de brazos, lo que lo convierte en la segunda mejor opción para escalas grandes de este hiperparámetro. Con el fin de determinar si *sparring* es superado por *Doubler* a escalas mucho mayores, se han hecho experimentos pasados los 200 brazos, llegando hasta cantidades superiores a 1000, pero la tendencia no varía.

Para comprobar si la distribución de recompensas de los brazos del entorno afecta significativamente al resultado, probamos a cambiarla por una de tipo Bernoulli con medias muestreadas aleatoriamente para cada brazo, de manera uniforme. Escogemos esta distribución por ser de relevancia en la práctica (por ejemplo, para modelar si a un usuario le gusta un artículo o no). Remarcamos que este cambio no varía las suposiciones estructurales: sigue habiendo transitividad y desigualdad triangular estocásticas, así como ganador Condorcet. El resultado del experimento se consigna en la Figura 3.3. En este caso, *DTS* funciona mejor que *Doubler* aunque la mejor política continúa siendo *sparring*. Al igual que en el caso gaussiano, otras políticas como *Interleaved Filter* y *Beat The Mean* se ven negativamente degradadas con el aumento del número de brazos, hasta el punto de igualar o empeorar el caso aleatorio.

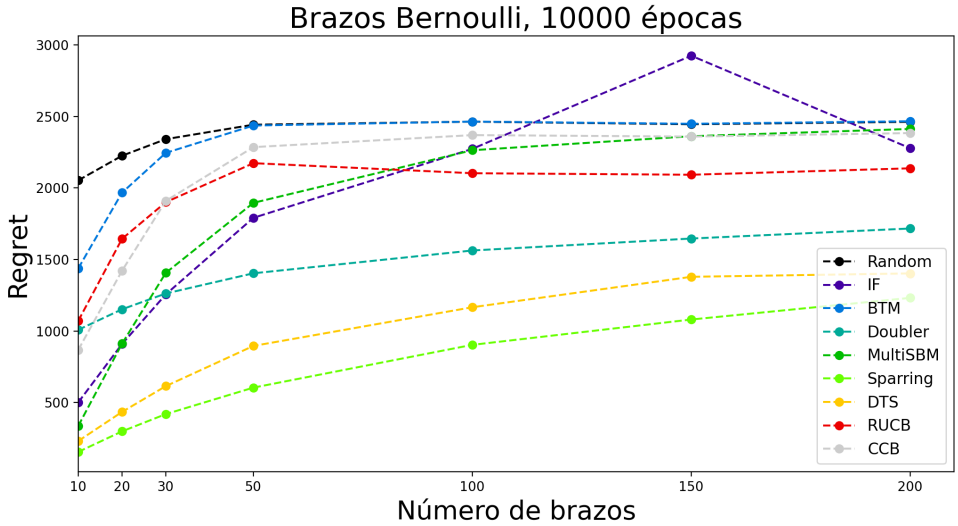

Figura 3.3: Regret acumulado para distinto número de brazos Bernoulli y $T = 10000$ épocas.

A continuación, comprobaremos cómo afecta la pérdida de suposiciones estructurales (falta de transitividad y de ganador Condorcet) al rendimiento de los agentes. Para ello, modificamos la distribución por parejas P_{ij} partiendo de la dada por los brazos gaussianos y modificándola posteriormente mediante un ruido aleatorio muestreado de una Gaussiana de media 0 y varianza d individualmente para cada par (i, j). Esto permitirá ajustar a voluntad el grado de falta de transitividad, es decir, de *ruido* en la distribución, modificando el valor de d. La razón por la que añadir ruido por pares rompe la

transitividad puede comprenderse atendiendo a las expresiones que la definen (Ecuación 2.9, 2.10): los términos que aparecen en las desigualdades están asociados a cada pareja de brazos y, como el ruido se aplica individualmente a cada pareja, cada uno de estos valores se ve modificado de manera independiente, lo que puede ocasionar que dejen de satisfacerse las desigualdades. Asimismo, añadir ruido da lugar a la posibilidad de que deje de existir ganador Condorcet. Realizamos el experimento para $N = 10$ brazos y $T = 4000$ épocas. El resultado se consigna en la Figura 3.4. Recordamos aquí por conveniencia que los únicos algoritmos que no asumen la existencia de ganador Condorcet en su diseño son *CCB* y *DTS* (cf. Tabla 2.3). Notablemente, y en coherencia con esta observación, a partir de cierto nivel de ruido el algoritmo *DTS* supera a *sparring* (hasta ahora imbatido en todos los escenarios) y se mantiene robusto al aumento de la varianza. Del mismo modo, mientras que *RUCB* se ve perjudicado por el nivel de ruido, *CCB* no se ve afectado aunque comparativamente continúa siendo peor que *sparring*.

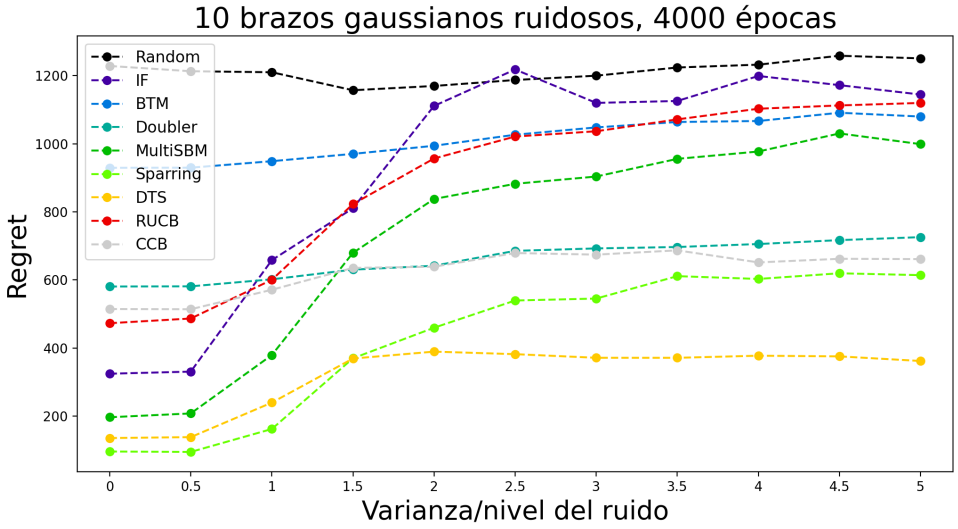

Figura 3.4: Regret acumulado para 10 brazos gaussianos con ruido por parejas y $T = 4000$ épocas.

Concluimos la sección volviendo al caso inicial de los brazos gaussianos y experimentando si la comparativa cambia al tomar como métrica el regret débil (ver la discusión final de la Subsección 2.2.1). El resultado se consigna en la Figura 3.5. Por ejemplo, el algoritmo *Doubler* se ve notablemente beneficiado en este caso dado que su política utiliza uno de los dos brazos de manera exploratoria y el otro para minimizar regret. Así, cuando solo se atiende a la opción de mejor regret, el resultado es mucho mejor que atendiendo a la media de los dos brazos. No obstante, *sparring* continúa siendo la mejor opción en este caso.

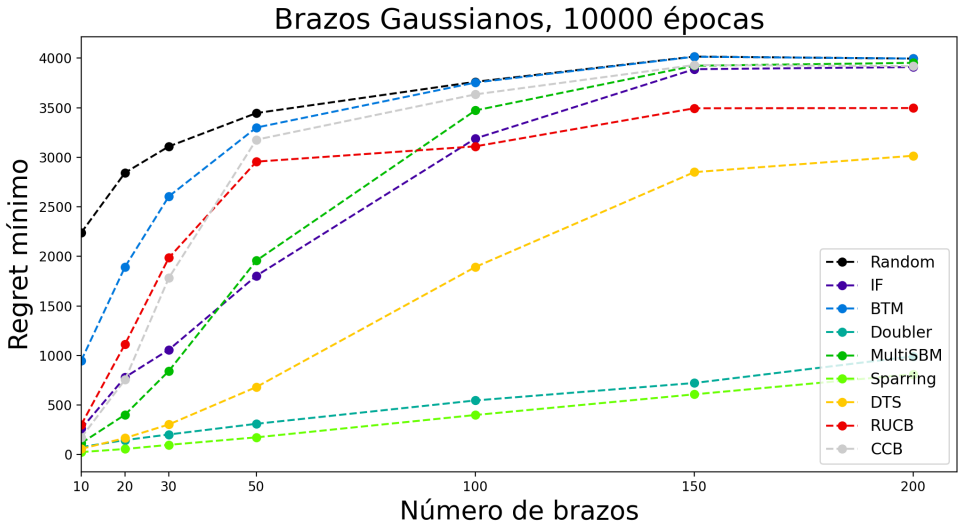

Figura 3.5: Regret mínimo acumulado para distinto número de brazos gaussianos y $T = 10000$ épocas.

3.4. Ajuste de parámetros

En la sección anterior se ha determinado que *sparring* es la mejor política para todas las situaciones probadas, a excepción del caso no transitivo sin ganador Condorcet, en el que *DTS* es preferible. Con el fin de maximizar el rendimiento obtenido y en vista de estos resultados, se seleccionan ambas políticas para ser optimizadas mediante el ajuste de sus hiperparámetros, cuyos resultados se presentan en esta sección. Comenzamos con *sparring*, cuyos hiperparámetros son los valores iniciales de α y β en el Thompson Sampling subyacente (denotados, como en la Subsección 2.1.4, por α_0 y β_0 respectivamente). Para optimizar el regret, realizamos una búsqueda en rejilla (*GridSearch*) combinando valores de los parámetros cuyo rango abarca desde valores muy bajos hasta muy altos. Los experimentos se realizan con $T = 10000$ épocas y para distintos valores del número de brazos, con el fin de detectar posibles variaciones dependiendo de la escala del problema. El resultado se muestra en la Figura 3.6. Observamos que la importancia de los hiperparámetros es mayor cuanto mayor es el número de brazos: en el caso $N = 5$ se consigue el mejor regret (inferior a 500) para muchas combinaciones de hiperparámetros, mientras que en el caso $N = 100$ la elección es mucho más reducida. En todos los casos, el mejor desempeño ocurre para valores de α_0 y β_0 cercanos a 1. Por ejemplo, la combinación $\alpha_0 = 1$, $\beta_0 = 10$ se encuentra entre las mejores para todos los rangos de brazos.

En el caso de *DTS*, un hiperparámetro a ajustar es γ. Recordamos del estudio teórico (cf. Subsección 2.2.6) que el algoritmo utiliza, al igual que *Thompson Sampling*, una distribución Beta subyacente cuyos parámetros fueron fijados a $\alpha_0 = \beta_0 = 1$ por los autores de [16] al ser suficientes para el análisis teórico. Sin embargo, en el ajuste de hiperparámetros realizado aquí los modificamos con el fin de optimizar el rendimiento real. Los experimentos se realizarán en el entorno en el que *DTS* es prin-

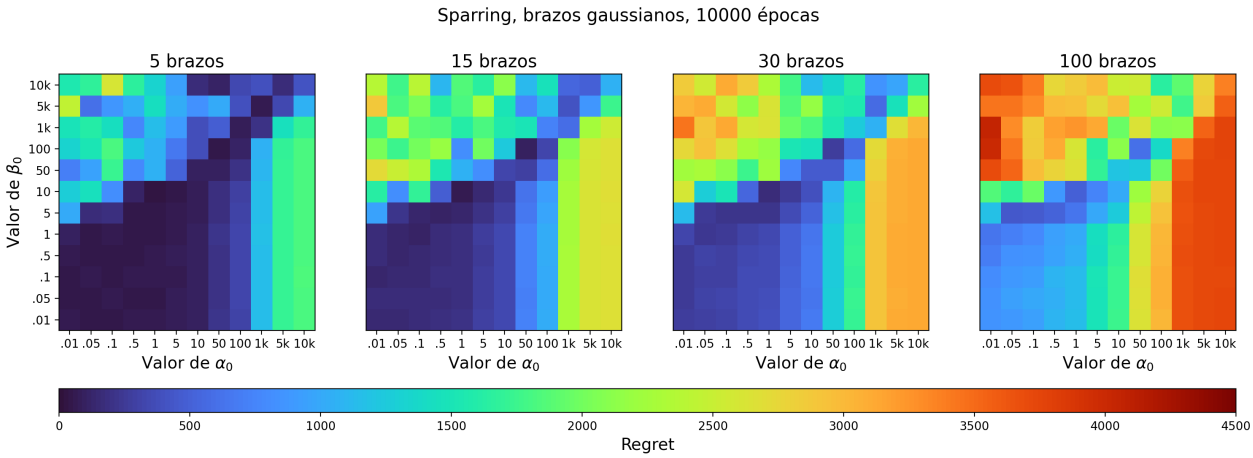

Figura 3.6: Ajuste de hiperparámetros para sparring.

cipal frente a sparring, es decir, añadiendo ruido por parejas a los brazos gaussianos para eliminar la transitividad. Fijamos la varianza del ruido $d = 2$, el número de brazos $N = 10$ y el número de épocas $T = 10000$. El resultado se muestra en la Figura 3.7. De todas las combinaciones analizadas, el óptimo sucede para $\alpha_0 = 0{,}05$, $\beta_0 = 0{,}01$, $\gamma = 1$, incurriendo en un regret de aproximadamente 500.

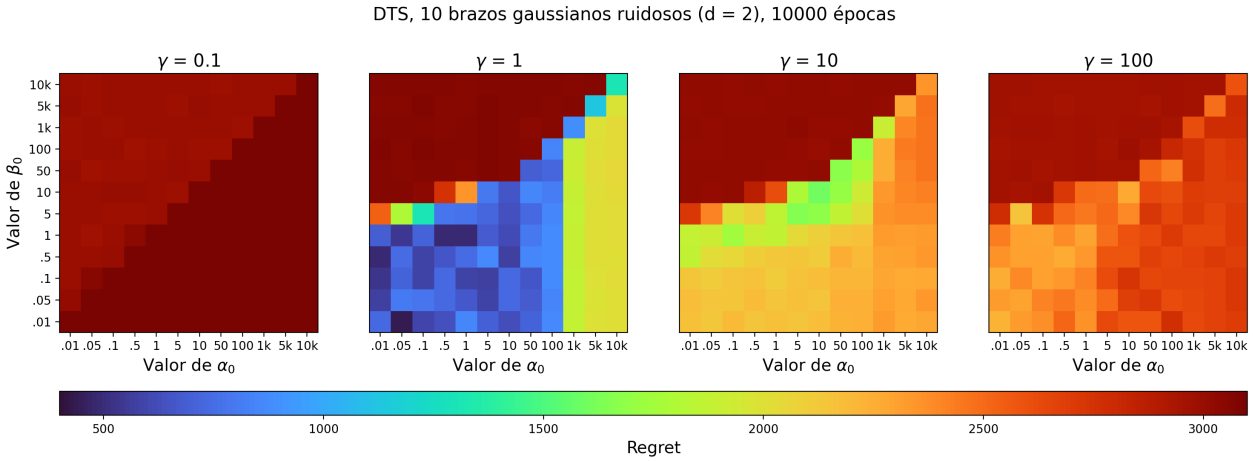

Figura 3.7: Ajuste de hiperparámetros para *DTS*.

3.5. Pérdida frente a bandidos multibrazo

Concluimos los análisis experimentales investigando las diferencias de regret entre las mejores políticas de bandidos duelistas encontradas en la sección anterior y los algoritmos de bandido multibrazo que tienen acceso a la recompensa numérica de la tirada realizada. Estos últimos deberían, a priori, alcanzar un rendimiento superior debido a la mayor cantidad de información de la que disponen, pero se busca comprobar si verdaderamente es así y en cuanta medida se diferencian. La motivación para esto es que los bandidos duelistas son más propicios a ser llevados a la práctica en escenarios

determinados en los que es más sencillo medir, en una situación dada, cuál de entre dos acciones es mejor (frente a obtener un valor numérico que indique la calidad de dicha acción). Un tal ejemplo es la realización de *tests A/B* entre dos sistemas, en los que la única realimentación que se mide como resultado del *test* es precisamente el resultado de una comparación. Para estos casos, es interesante conocer si se está perdiendo demasiada eficiencia al prescindir de la recompensa numérica o, por el contrario, convendría invertir esfuerzo en conseguir otras metodologías fiables que permitan extraer valores cardinales para cada acción.

Para este experimento, fijamos $T = 1000$ épocas y variamos el número de brazos, registrando el *regret* final. Enfrentaremos a los mejores bandidos duelistas encontrados anteriormente con los algoritmos de bandido multibrazo: *Thompson Sampling* con distribución *a priori* gaussiana (al ser las recompensas gaussianas), *UCB* con $\gamma = 0{,}1$, ε-*greedy* con $\varepsilon = 0{,}1$ y *EXP3* con el γ óptimo según la expresión ofrecida en [4]. Aunque no se ha desarrollado detalladamente *EXP3* en este trabajo, como se indica en la Subsección 2.1.5, lo incluimos en el experimento por completitud. Asimismo, añadimos *sparring* con agente subyacente *UCB* con $\gamma = 0{,}1$ para comprobar esta variante frente a *Thompson Sampling*. El valor de $\gamma = 0{,}1$ para *UCB* ha sido seleccionado probando varias alternativas y escogiendo la de menor regret. Los agentes multibrazo solo accionan un brazo en cada ronda, en lugar de una pareja, y tienen acceso al valor de recompensa resultado de la acción. A efectos del cálculo del regret, se considera que se ha accionado una pareja cuyos elementos son el brazo seleccionado.

El resultado del experimento se muestra en la Figura 3.8. En este rango de épocas y con brazos gaussianos, los algoritmos que mejor desempeñan son los sparring y *UCB*. Al tener la información numérica, *UCB* mejora a sparring en una cantidad estable con respecto al número de brazos. Para valores de brazos grandes, ε-greedy supera a sparring, en parte debido al número reducido de épocas en comparación.

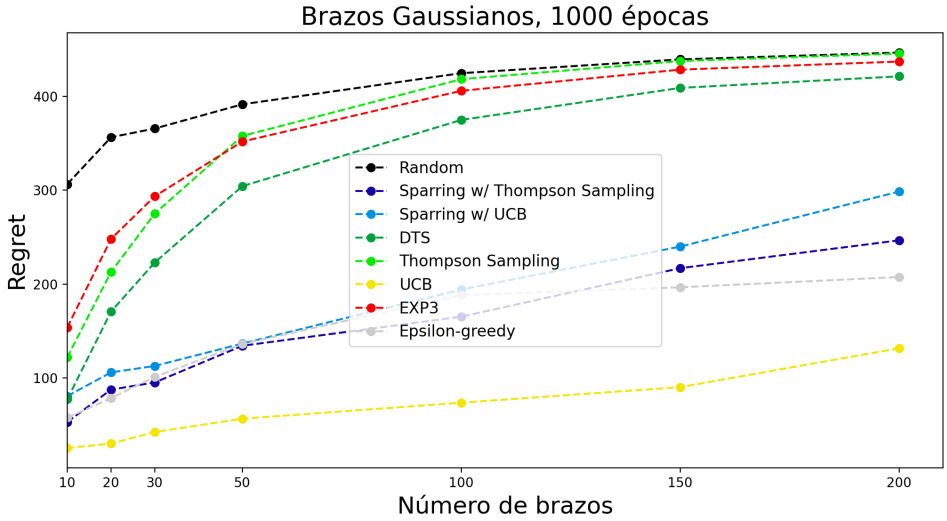

Figura 3.8: Regret acumulado contra bandidos multibrazo.

4

Conclusiones

El problema de los bandidos duelistas es una variante del bandido multibrazo, necesaria a causa de su presencia en determinadas situaciones prácticas, como los *tests A/B*, que son más sencillas de modelar empleando comparaciones por parejas que extrayendo puntuaciones numéricas individuales. Con el fin de realizar un estudio detallado del problema, se han recopilado y presentado en primer lugar los principales algoritmos para los agentes presentes en la literatura de investigación sobre el problema, que siguen diversas estrategias y asumen distintas restricciones sobre el problema.

Una vez determinada la selección de algoritmos a estudiar, se han propuesto distintos objetivos en forma de preguntas a responder acerca del rendimiento de los mismos. Con el fin de atacar estos objetivos, y ante la falta de implementaciones existentes con la suficiente flexibilidad como para comparar todos los agentes elegidos en las situaciones planteadas, se ha desarrollado un marco de simulación con bandidos duelistas en el que se han implementado los algoritmos deseados. Con ello, se han realizado los experimentos que permiten responder a las preguntas planteadas.

En primer lugar, se ha determinado que la política conocida como **sparring** resulta ganadora en cuanto a rendimiento (menor cantidad de regret incurrido) en la mayoría de situaciones probadas, incluyendo distintos entornos (gaussianos y Bernoulli), distintos tipos de regret (promedio y mínimo), con enorme robustez al incremento del número de brazos (manteniéndose vencedor tanto en cantidades bajas como en cantidades altas) y a la escala temporal. Esto pone de manifiesto la necesidad de usar esta política en casos reales, a pesar de la falta de garantías teóricas establecidas sobre la misma. Por otra parte, mediante el uso de entornos con ruido añadido para romper ciertas suposiciones estructurales sobre la distribución por pares, como la transitividad y la existencia de ganador único (o Condorcet), se ha comprobado que sparring no es robusto ante la falta de estas suposiciones, empeorando su rendimiento y siendo superado por **Double Thompson Sampling (DTS)**, en cuyo diseño se tienen en cuenta estas situaciones. De este modo, en situaciones con baja transitividad se pone de manifiesto la necesidad de reemplazar sparring por DTS para un rendimiento óptimo.

Una vez establecido que los candidatos de interés son sparring y DTS, se ha analizado el efecto de sus hiperparámetros sobre el rendimiento de las políticas. En el caso de sparring, se ha determinado que, fijado el número de épocas, el efecto es poco significativo cuando el número de brazos es

pequeño pero puede ocasionar diferencias significativas de regret para cantidades grandes de brazos. Por otra parte, en el caso de DTS se ha comprobado que es importante realizar un ajuste de hiperparámetros dado que, dependiendo de la configuración de los tres hiperparámetros involucrados, se obtienen valores de regret muy distintos. En ambos casos, mediante los experimentos de tipo *Grid-Search* realizados, se han determinado configuraciones de parámetros óptimas para las situaciones analizadas.

Finalmente, se ha estudiado cómo afecta la falta de recompensas numéricas en el rendimiento, comparándolos directamente con agentes para el problema del bandido multibrazo original, que tienen acceso a dichos valores de recompensa. Se ha determinado en la situación estudiada que, para una cantidad de épocas y brazos pequeña, el mejor algoritmo multibrazo (UCB) supera a sparring y, conforme aumenta el número de brazos, otros algoritmos multibrazo también lo hacen, poniendo de manifiesto una cierta penalización en el rendimiento causada por no conocer directamente los valores de recompensa.

4.1. Trabajo futuro

Como trabajo futuro, en base a las conclusiones extraídas de este trabajo, se propone:

1.– En vista de sus buenos resultados, realizar un análisis teórico de sparring que permita establecer garantías sobre sus resultados con la menor cantidad de suposiciones posible. Hasta ahora no existe tal análisis y es posible que a través del mismo puedan encontrarse puntos de mejora que lo hagan estable a la falta de transitividad o que determinen qué condiciones debe cumplir el agente multibrazo subyacente para optimizar el rendimiento.

2.– Comparar los agentes en experimentos en situaciones reales con usuarios. Aunque es más difícil y laborioso realizar todas las pruebas que se han llevado a cabo de manera simulada en entornos reales, podrían hacerse experimentos reales en alguna situación concreta para establecer si los resultados simulados son extrapolables a datos reales.

3.– Diseñar metodologías para determinar qué suposiciones (transitividad, existencia de ganador único) se dan en distintos entornos reales y aplicarlas para decidir si en casos prácticos es preferible aplicar sparring o DTS.

4.– Estudiar generalizaciones aún más recientes del problema, como las presentadas en la Subsección 2.2.8, con el fin de responder preguntas similares a las planteadas aquí en el caso de los bandidos duelistas.

Bibliografía

[1] N. Ailon, Z. Karnin, and T. Joachims, "Reducing dueling bandits to cardinal bandits," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, p. II–856–II–864.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, pp. 235–256, 05 2002.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire, "Gambling in a rigged casino: The adversarial multi-armed bandit problem," *Foundations of Computer Science, 1975., 16th Annual Symposium on*, 07 1998.

[4] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002.

[5] B. Brost, I. J. Cox, Y. Seldin, and C. Lioma, "An improved multileaving algorithm for online ranker evaluation," *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016.

[6] R. Busa-Fekete, B. Szorenyi, W. Cheng, P. Weng, and E. Huellermeier, "Top-k selection based on adaptive sampling of noisy preferences," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1094–1102.

[7] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011.

[8] Y. Du, S. Wang, and L. Huang, "Dueling bandits: From two-dueling to multi-dueling," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, p. 348–356.

[9] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020.

[10] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay, "Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search," *ACM Trans. Inf. Syst.*, vol. 25, no. 2, p. 7–es, apr 2007.

[11] T. Lu, D. Pal, and M. Pal, "Contextual multi-armed bandits," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning

Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 485–492.

[12] S. Y. Ramamohan, A. Rajkumar, S. Agarwal, and S. Agarwal, "Dueling bandits: Beyond condorcet winners to general tournament solutions," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[13] A. Saha, T. Koren, and Y. Mansour, "Adversarial dueling bandits," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 9235–9244.

[14] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018.

[15] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Machine Learning: ECML 2005*, J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 437–448.

[16] H. Wu and X. Liu, "Double thompson sampling for dueling bandits," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[17] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims, "The k-armed dueling bandits problem," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1538–1556, 2012, jCSS Special Issue: Cloud Computing 2011.

[18] Y. Yue and T. Joachims, "Interactively optimizing information retrieval systems as a dueling bandits problem," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1201–1208.

[19] ——, "Beat the mean bandit," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Madison, WI, USA: Omnipress, 2011, p. 241–248.

[20] M. Zoghi, Z. S. Karnin, S. Whiteson, and M. de Rijke, "Copeland dueling bandits," *CoRR*, vol. abs/1506.00312, 2015.

[21] M. Zoghi, S. Whiteson, R. Munos, and M. Rijke, "Relative upper confidence bound for the k-armed dueling bandit problem," *31st International Conference on Machine Learning, ICML 2014*, vol. 2, 12 2013.

Apéndices

A

Framework de simulación con bandidos duelistas

Con el fin de llevar a cabo los experimentos simulados deseados en el estudio del problema de los bandidos duelistas, se ha implementado y documentado una librería para comparar distintas políticas para agentes que se enfrentan al problema de los bandidos duelistas [18]. La librería está implementada utilizando en todo momento las estructuras y operaciones vectorizadas de *NumPy* [9] para una mayor eficiencia. Asimismo, ha sido optimizada en la medida de lo posible con la ayuda del *profiler cProfile* (https://docs.python.org/3/library/profile.html#module-cProfile). El código documentado puede encontrarse en https://github.com/MiguelGonzalez2/bandits.

Las funcionalidades principales soportadas son:

1.– Implementación de políticas para agentes del problema de bandido multibrazo (MAB) y de los bandidos duelistas (DB).

2.– Implementación de entornos para bandidos multibrazo (MAB) y bandidos duelistas (DB).

3.– Definición y ejecución de experimentos, en los que un conjunto deseado de agentes se enfrenta a un entorno predefinido durante el número de épocas establecido y que se repite el número de veces indicado.

4.– Almacenamiento de distintas métricas que permiten evaluar el rendimiento de cada agente en cada experimento.

5.– Definición y ejecución de *simulaciones*, que son grupos lógicos de experimentos que se ejecutan secuencialmente y cuyos resultados se almacenan de manera conjunta.

6.– Generación (utilizando *PyPlot* (https://matplotlib.org/stable/tutorials/introductory/pyplot.html) de *MatPlotLib*) de gráficas individuales por experimento y agregadas por simulación.

La librería se divide en tres módulos:

1.– *agents*: Implementa lo relativo a los agentes para MABs y DBs. Las clases principales que aporta este módulo son *DBAgent* y *MABAgent*, de las cuales puede heredarse para definir agentes personalizados como se explica posteriormente.

2.– *environments*: Implementa lo relativo a los entornos para MABs y DBs. La clase principal que aporta este módulo es *Environment*, de la cual puede heredarse para definir entornos personalizados como se explica posteriormente.

3.– *simulation*: Implementa lo relativo a la simulación, almacenamiento de métricas y generación de gráficas.

La estructura se resume en el diagrama de la Figura A.1.

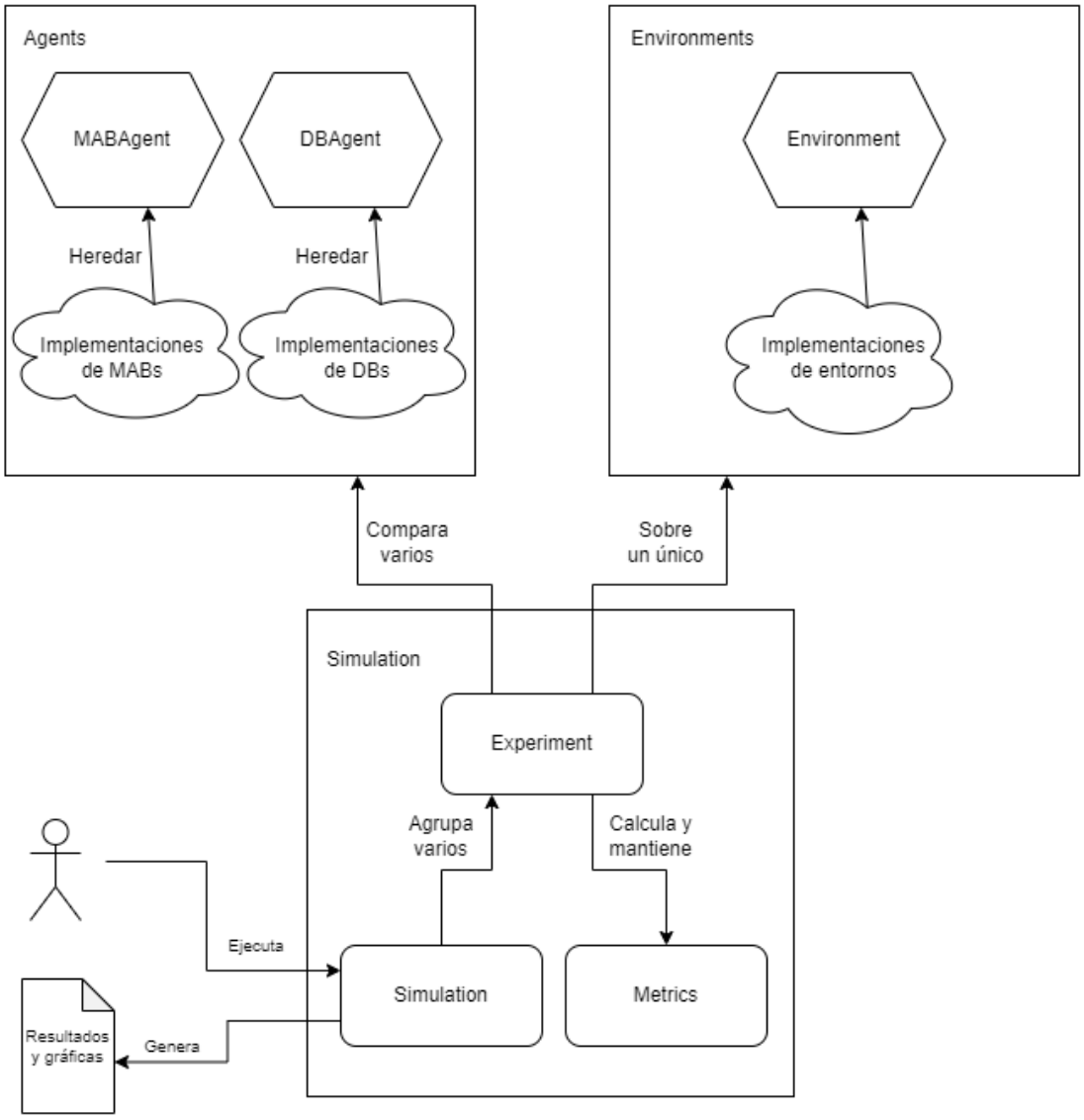

Figura A.1: Diagrama ilustrativo de la estructura general de la librería.

Para definir un nuevo agente DB (respectivamente, MAB), basta con crear una clase que herede de *DBAgent* (respectivamente *MABAgent*) y sobrescriba los métodos indicados en la documetación de la librería. Se incluyen implementaciones de los siguientes agentes MAB: ε*-greedy, UCB, Thompson Sampling* con distribución beta, *Thompson Sampling* con distribución Gaussiana y *EXP3*. Asimismo, se incluyen implementaciones de los siguientes agentes DB: aleatorio, *Interleaved Filter*, *Beat The Mean*, *Sparring*, *Doubler*, *MultiSBM*, *RUCB*, *CCB* y *DTS*.

Para definir un entorno para los agentes MAB y DB, basta con crear una clase que herede de *Environment* y sobrescriba los métodos indicados en la documetación de la librería. Se incluyen implementaciones de los siguientes entornos: brazos *Bernoulli*, brazos *gaussianos*, brazos *gaussianos ruidosos* con ruido aleatorio en las distribuciones por parejas y brazos con distribución *piedra-papel-tijeras*.

Para llevar a cabo una simulación, basta con crear un objeto *Simulation* suministrándole los objetos *Experiment* deseados. La librería incluye métodos para ejecutar la simulación, guardar y cargar el estado intermedio de la misma y generar gráficas con distintas métricas, de entre las siguientes: recompensa media, regret estándar (puntual y acumulado), regret Copeland (puntual y acumulado), regret Copeland débil o fuerte y porcentaje de elecciones óptimas.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID