Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 21 (2024) 328-338

www.revista-riai.org

Optimización Bayesiana no miope POMDP para procesos con restricciones de operación y presupuesto finito

José Luis Pitarch [a,*], Leopoldo Armesto [b], Antonio Sala [a]

[a] *Instituto de Automática e Informática Industrial (ai2), Universitat Politècnica de València, Camino de Vera, s/n, 46022, Valencia, España.*
[b] *Instituto de Diseño y Fabricación (IDF), Universitat Politècnica de València, Camino de Vera, s/n, 46022, Valencia, España.*

To cite this article: Pitarch, J.L., Armesto, L., Sala, A. 2024. POMDP non-myopic Bayesian optimization for processes with operation constraints and a finite budget. Revista Iberoamericana de Automática e Informática Industrial 21, 328-338. https://doi.org/10.4995/riai.2024.21142

Resumen

Mejorar la toma de decisiones a partir de los resultados observados tras la experimentación es una tarea habitual en muchas aplicaciones, tanto a nivel de investigación en laboratorio como en procesos de producción industriales. Sin embargo, realizar experimentos suele acarrear un coste no despreciable, por lo que una excesiva exploración es perjudicial. La optimización bayesiana es una técnica muy utilizada en este contexto, decidiendo la siguiente experimentación en base a un modelo estadístico. No obstante, está técnica no tiene en cuenta explícitamente el coste real de realizar un experimento, ni si existe un presupuesto (o número de experimentos, tiempo, etc.) máximo. El problema de toma de decisiones bajo incertidumbre y presupuesto finito puede plantear como un Proceso de Decisión de Márkov Parcialmente Observable (POMDP, por sus siglas en inglés). Este trabajo aborda el problema de optimización experimental sujeta a restricciones de operación con un enfoque POMDP, donde las posibles decisiones vienen proporcionadas por heurísticas de la optimización bayesiana, o de otra índole definida por el usuario. La estrategia consiste en construir un árbol de posibles escenarios partir del conocimiento (incierto) acerca del proceso/sistema aprendido a partir de experimentos previos. Dicho conocimiento se modela mediante procesos Gaussianos, que se actualizan con cada nueva observación. La evaluación sobre la mejor decisión a tomar se realiza mediante programación dinámica. El algoritmo desarrollado ha sido evaluado mediante comparación con otras opciones de la literatura en un banco de pruebas sintético, y para optimizar un proceso químico de producción por lotes.

Palabras clave: Programación dinámica, Optimización de procesos, Procesos Gaussianos, Optimización bajo incertidumbre.

POMPD non-myopic Bayesian optimisation for processes with operation constraints and a finite budget.

Abstract

Improving decision making from the observed results after experimentation is a usual task in many applications, from the research laboratory scale to industrial production systems. However, conducting experiments often takes a non-negligible cost. Consequently, excessive exploration is harmful. Bayesian optimization is a widely used technique in this context, deciding next experiment based on a statistical model. However, this technique does not explicitly account for the actual cost of the experiment, nor whether a limited budget (economic, number of experiments, time, etc.) exists. The problem of decision making under uncertainty and finite sample budget can be cast as a Partially Observable Markov Decision Process (POMDP). This work addresses the experimental optimization problem with operation constraints by a POMDP approach, where the possible actions to make are given by well-known Bayesian optimization heuristics, or any other defined by the user. The strategy consists in building a scenario tree from the (uncertain) knowledge about the system/process, learnt from prior experiments. Such a knowledge is modelled by Gaussian processes, which are updated with each new available observation. The evaluation on the best action to make is realized via dynamic programming. The developed algorithm has been evaluated by comparison with other options in the literature in a synthetic test bench, and to optimize a chemical batch production process.

Keywords: Dynamic programming, Process optimisation, Gaussian processes, Optimisation under uncertainty.

*Autor para correspondencia: jlpitarch@isa.upv.es

1. Introducción

El ajuste de modelos o la optimización a través de la experimentación-observación es una tarea a llevar a cabo en cualquier proceso de toma de decisiones donde no se dispone de un conocimiento perfecto del sistema y/o el entorno, es decir, no se dispone de un buen modelo de predicción. Existen diversas técnicas aplicables en este contexto: las indirectas, basadas en identificación de modelos y control adaptativo (Yip y Marlin, 2003; Rodríguez-Blanco et al., 2017); las directas, sin uso de modelos, basadas en la búsqueda y aprendizaje de la política de control óptima (Deisenroth et al., 2013).

En problemas donde realizar un excesivo número de experimentos reales acarrea un coste importante, se utilizan técnicas indirectas que buscan el óptimo sobre un modelo subrogado, previo paso al experimento real. La optimización bayesiana (OB) es la técnica más popular en este contexto (Frazier, 2018; Calandra et al., 2016; del Rio Chanona et al., 2021; Mora et al., 2023), donde el conocimiento (incierto) de la función a optimizar se suele caracterizar de forma probabilística por un proceso gaussiano (PG). Los procesos gaussianos son aproximadores no paramétricos basados en datos, aunque también permiten componentes paramétricos en su modelo, incorporando en ellos la regresión clásica (Rasmussen y Williams, 2006; Wu y Movellan, 2012). A partir del modelo probabilístico de la función objetivo como un PG, la OB se basa en optimizar una cierta *función de adquisición,* que es una heurística con un cierto compromiso entre exploración y explotación. Existen varias funciones de adquisición disponibles en la literatura (Frazier, 2018). No obstante, la OB está pensada para resolver un problema de optimización estático y, en su formulación, no considera explícitamente el coste acumulado de experimentos, ni que pueda existir un límite en número de estos.

Encontrar las decisiones que arrojen el mejor compromiso exploración-explotación bajo incertidumbre, cuando realizar un experimento acarrea un coste, pero a su vez puede mejorar el conocimiento del sistema, convierte el problema de optimizar una función estática en un problema dinámico. Este tipo de problemas se presentan en la literatura como procesos de decisión de Márkov parcialmente observables, POMDP de sus siglas en la literatura en lengua inglesa (Spaan, 2012). La OB aplicada a éstos podría considerarse sólo como una heurística *miope* de predicción a un paso (Astudillo et al., 2021).

Un problema POMDP debe ser resuelto mediante programación dinámica (PD), de forma que estime las potenciales consecuencias de tomar decisiones a varios pasos futuros (Busoniu et al., 2017; Armesto y Sala, 2022). Sin embargo, resolver estos problemas de forma óptima suele ser intratable computacionalmente y, en especial, cuando las variables de decisión pueden tomar infinitos valores en un espacio continuo.

Una idea interesante es combinar OB y el enfoque POMDP para obtener algoritmos no miopes, i.e. capaces de estimar el coste (i.e. la función de valor) a varios pasos futuros simulando (y actualizando) el PG con decisiones dadas por una función de adquisición de OB prefijada (Lam et al., 2016; Astudillo et al., 2021; Paulson et al., 2022); este tipo de enfoque combinado se denotará en este trabajo con las siglas OB-POMDP. No obstante, la aplicación directa de esta idea requiere resolver un costoso problema de optimización anidada. Es por ello que las propuestas de la literatura que tienen un enfoque más práctico limitan la búsqueda de decisiones en el espacio continuo a la primera etapa y/o la estimación del coste a dos pasos únicamente (Wu y Frazier, 2019).

En este trabajo se aborda la optimización experimental pura (i.e., sin disponibilidad de modelos fenomenológicos) con un presupuesto finito como un problema de tipo OB-POMDP. Para ello, proponemos una estrategia no miope, pero más eficiente computacionalmente que las alternativas que implican optimización anidada en la literatura. La propuesta se inspira en la idea de Hoffman et al. (2011) sobre poder evaluar de forma dinámica (i.e., en cada etapa de predicción elige entre) varias funciones de adquisición. El problema en el citado trabajo es que la evaluación se realiza basada en el coste/ganancia acumulado en experimentos pasados, mientras que lo correcto desde el punto de vista de la programación dinámica es explorar el efecto de decisiones futuras. Esto es precisamente lo que proponemos en el presente trabajo.

Cabe resaltar que nosotros buscamos la aplicación de estas técnicas OB-POMDP a procesos de carácter industrial o experimental donde el presupuesto disponible representa un factor crítico. Para demostrarlo, aplicamos el algoritmo desarrollado a resolver un problema de optimización experimental en un reactor químico de producción por lotes, donde cada lote producido consume recursos y el beneficio económico total de producción se obtiene una vez se ha cumplido la producción de todos los lotes demandados. Una versión preliminar de los resultados aquí presentados aparece en trabajos de conferencia (Armesto et al., 2023; Pitarch et al., 2023). En el primero, la estrategia de optimización aquí propuesta se testeó en un banco de pruebas con funciones a optimizar generadas de forma sintética, a partir de un PG base. En el segundo se aplicó la idea para optimizar el mencionado problema del reactor químico, pero sin existencia de restricciones de proceso, más allá de meros límites en las variables de decisión. Evidentemente, estas no son las condiciones existentes en un caso de aplicación real.

Por ello, en este trabajo extendemos la propuesta al caso general con restricciones, y lo aplicamos al proceso químico con existencia de restricciones duras en los productos generados. Asimismo, en este artículo se adjunta el pseudocódigo detallado sobre las clases y métodos necesarios para implementar nuestra propuesta, incluyendo su posible paralelización en múltiples núcleos de procesamiento.

El resto del artículo se estructura como sigue. La siguiente sección resume los conceptos de regresión con PGs, OB con restricciones, y formaliza el problema POMDP a resolver. La Sección 3 presenta una síntesis del algoritmo OB-POMDP desarrollado y una evaluación comparativa frente a algunas propuestas pertinentes de la literatura. En la Sección 4 se presentan y discuten los resultados obtenidos en el caso de estudio del reactor químico. Finalmente, el artículo se cierra dando conclusiones y perspectivas de aplicación. El pseudocódigo se presenta en un apéndice al final del trabajo.

2. Conceptos preliminares y problema a resolver

Formalmente, el objetivo de la OB es encontrar el óptimo $x^* = \arg\min_{x\in\mathbb{X}} f(x)$, $f: \mathbb{X} \to \mathbb{R}$, donde $\mathbb{X}$ denota el conjunto de decisiones válidas, sin asumir existencia de dinámica en f. Es decir, f devuelve el desempeño observado de realizar un experimento costoso con unos ciertos valores de las variables de decisión x. Para ello, se optimiza alguna heurística con un modelo probabilístico de f, generalmente un Proceso Gaussiano.

Procesos Gaussianos. La formalización de una "función aleatoria" se suele hacer mediante un PG, que se caracteriza por una media $\mu(x)$ y un generador de covarianza o "kernel" $\text{cov}\big(f(x_a), f(x_b)\big) \coloneqq \kappa(x_a, x_b)$, véase Rasmussen y Williams (2006). Conforme se adquieren datos, el modelo probabilístico de la función objetivo se actualiza. Por ello, diferenciaremos entre el PG a *priori* y a *posteriori* una vez se vayan hayan incorporado acciones X y sus correspondientes medidas Y al PG:

$$X \coloneqq [x_0, x_1, x_2, \dots, x_n]; \quad Y \coloneqq [y(x_0), y(x_1), \dots, y(x_n)];$$

donde $y(x) = f(x) + w, \ w \approx \mathcal{N}(0, \lambda^2)$, siendo λ la desviación típica del ruido de medida que suponemos asociado a las observaciones de f.

La media y "kernel" del PG a *posteriori* se calculan como:

$$\mu_{\mathcal{B}}(x) = \mu(x) + \kappa(x, X)(\kappa(X, X) + \lambda^2 I)^{-1}\big(Y - \mu(X)\big) \quad (1)$$

$$\kappa_{\mathcal{B}}(x_a, x_b) = \kappa(x_a, x_b) - \kappa(x_a, X) \cdot (\kappa(X, X) + \lambda^2 I)^{-1}\kappa(x_b, X)^T \quad (2)$$

Siendo $\mathcal{B} \coloneqq \{X, Y\}$ y $\mu(x)$ y $\kappa(x_a, x_b)$ la media y "kernel" del PG prior (Rasmussen y Williams, 2006). La varianza marginal de la predicción de un PG en x es $\Sigma_{\mathcal{B}}(x) \coloneqq \kappa_{\mathcal{B}}(x, x)$. Aunque estrictamente $\mathcal{B}$ está definido como el conjunto de datos para el cálculo del PG posterior, en ocasiones denotaremos a un PG como $\mathcal{B}$ por brevedad en la notación, si la media y "kernel" están claros por el contexto.

Los PGs pueden contener una parte paramétrica, de forma que su función de correlación se puede expresar como:

$$\kappa(x_a, x_b) \coloneqq \bar{\kappa}(x_a, x_b) + \phi(x)\Sigma_\theta \phi(x)^T \quad (3)$$

donde $\bar{\kappa}(x_a, x_b)$ es un generador de covarianza no-paramétrico y θ un conjunto de parámetros ajustables, tal que $\phi(x)\theta$ es la predicción de la parte paramétrica y Σ_θ es la matriz de covarianzas a priori de los parámetros (Wu y Movellan, 2012). Si se desea conocer, el valor estimado de los parámetros es:

$$\hat{\theta} = \Sigma_\theta \phi(X)^T (\kappa(X, X) + \lambda^2 I)^{-1} \quad (4)$$

Este trabajo asume que no se dispone de un modelo de primeros principios del proceso a optimizar, por tanto $\mu(x) = 0$. Además, θ se restringe a estimar una constante o "bias" del PG, haciendo $\phi(x) = 1$. Este método, cuando $\Sigma_\theta \to \infty$, es conocido como *ordinary krigging* en la literatura científica, y es una técnica ampliamente utilizada para diseño óptimo de experimentos, e.g., en geología matemática (Cressie, 1990), optimización automática de dispositivos de control de flujo (Duvigneau y Chandrashekar, 2012), entre otras aplicaciones.

2.1. Optimización Bayesiana estándar

Dado un PG, definido por su media (1), generador de convarianzas (2), y un conjunto de datos históricos $\mathcal{B}$, los algoritmos de OB determinan el mejor experimento en base a una función subrogada $F(\mathcal{B}, x)$. Dicha función subrogada, conocida en la literatura como función de adquisición, debe tener un coste de evaluación despreciable frente al coste de experimentación, i.e., la evaluación de la verdadera función objetivo f. Su propósito es guiar la optimización en base a una serie de heurísticas que implican un cierto compromiso explotación/exploración. De entre las funciones de adquisición más conocidas, podemos encontrar (Frazier, 2018):

- Valor esperado (EV, *Expected Value*): $\mu_{\mathcal{B}}(x)$.
- Probabilidad de mejorar (PI, *Probability of Improvement*): $\frac{1}{2}\left(1 + \text{erf}\left(\frac{\mu_y(x^*) - \mu_{\mathcal{B}}(x)}{\sqrt{2\Sigma_{\mathcal{B}}(x)}}\right)\right)$

  Siendo $x^* \in X$ la mejor decisión evaluada hasta el momento y μ_y la media de las observaciones en dicho punto.
- Mejora esperada (EI, *Expected Improvement):* $\mathbb{E}\{\tilde{I}(x)\big|x\}$, siendo $\tilde{I}(x) = \max\Big(0, \mu_y(x^*) - y(x)\Big)$.
- Límite de confianza inferior/superior (LCB/UCB, *Lower/Upper Confidence Bound):*

$$F_\delta(\mathcal{B}, x) \coloneqq \mu_{\mathcal{B}}(x) \pm \eta\sqrt{\Sigma_{\mathcal{B}}(x)} \quad (5)$$

  tal que $1 - \delta = \text{erf}\left(\frac{\eta}{\sqrt{2}}\right)$.

El algoritmo básico consiste en inicializar el PG con un histórico de datos experimentales $\mathcal{B}$ y determinar el punto para realizar la siguiente experimentación según el criterio:

$$x = \arg\min_{x \in \mathbb{X}} F(\mathcal{B}, x), \text{ o bien, } x = \arg\max_{x \in \mathbb{X}} F(\mathcal{B}, x)$$

Seguidamente se debe actualizar el histórico de datos con la observación experimental obtenida:

$$\mathcal{B} \leftarrow \mathcal{B} \oplus \{x, y(x)\} = \{X \cup x, Y \cup y(x)\}$$

Este proceso se repite hasta cumplir cierto criterio de finalización, por ejemplo, el número de experimentos o tiempo de ejecución límite.

2.2. Optimización Bayesiana con restricciones

Para el caso de existir restricciones en el sistema a optimizar, además de la función de desempeño $f(x)$ debemos considerar un conjunto de funciones $g(x) < 0$. Obviamente, si las restricciones sólo son en el espacio de decisiones, es trivial incorporarlas en la definición del conjunto $\mathbb{X}$, con lo que estaríamos en el caso estándar antes mencionado. Sin embargo, si $g(x)$ son (parcialmente) desconocidas, pero se dispone de medidas acerca de ellas, entonces se puede asociar dicha información a un segundo PG que, ante la dificultad de inferir la correlación a priori entre f y g en la práctica, supondremos independiente del PG que modela la función objetivo f.

Por tanto, en OB con restricciones sujetas a incertidumbre, el conjunto de datos históricos estará compuesto por las decisiones x utilizadas en la experimentación y las observaciones obtenidas tanto de la función f como de las restricciones g, manteniendo un conjunto $\mathcal{B}$ de datos separados para cada PG:

$$\mathcal{B}_f = \big\{[x_0, x_1, \dots], [y_f(x_0), y_f(x_1), \dots]\big\}$$

$$\mathcal{B}_g = \big\{[x_0, x_1, \dots], [y_g(x_0), y_g(x_1), \dots]\big\}$$

siendo $y_f(x) = f(x) + w_f$, $y_g(x) = f(x) + w_g$, con w_f y w_g dos ruidos independientes Gaussianos de desviación típica λ_f y λ_g, respectivamente. Las medias y funciones de covarianza *a posteriori* de los PGs de f y g, se denotarán como: $\mu_{\mathcal{B}_f}$, $\mu_{\mathcal{B}_g}$; $\kappa_{\mathcal{B}_f}$, $\kappa_{\mathcal{B}_f}$, respectivamente.

Una aproximación para resolver el problema de optimización restringida con incertidumbre es condicionar la mejora de desempeño esperada a la probabilidad de cumplir las restricciones (Gardner et al., 2014):

$$F(\mathcal{B}, x) \coloneqq \mathbb{E}\{\tilde{I}(x)\big|x\} \cdot \mathbb{P}(g(x) < 0)$$

Sin embargo, esto no garantiza el cumplimiento de las restricciones, y por tanto debe aplicarse en aquellos supuestos en los que está permitido "fallar". Por el contrario, si el incumplimiento de las restricciones es especialmente lesivo en una aplicación concreta, otra aproximación más adecuada consiste en

garantizar el cumplimiento de las restricciones con probabilidad $1-\delta$, por ejemplo, restringiendo la optimización a cumplir el límite de confianza superior/inferior (5) del PG asociado a las restricciones g (Girbés-Juan et al., 2023). Nótese que, si la restricción es incierta con distribución Gaussiana, no puede garantizarse al 100% la satisfacción de restricciones, pero nos podemos aproximar haciendo δ suficientemente pequeño (Jaiswal et al., 2023). Por ejemplo, un valor de $\eta = 3$ desviaciones típicas implica una probabilidad teórica de violar restricciones del 0.27%, que en la práctica se asocia a una certeza "casi" absoluta de no violar g (un fallo cada 370 pruebas).

En este segundo caso, la OB determina que la siguiente experimentación debe realizarse en base al siguiente criterio:

$$x = \arg\min_{\substack{x\in\mathbb{X} \\ G_\delta(\mathcal{B}_g,x)<0}} F(\mathcal{B},x), \text{ o bien, } x = \arg\max_{\substack{x\in\mathbb{X} \\ G_\delta(\mathcal{B}_g,x)<0}} F(\mathcal{B},x)$$

Siendo F una función de adquisición dada y $G_\delta(\mathcal{B}_g, x)$ el adecuado límite de confianza superior del PG que modela g (i.e., F_δ en (5) es la referida G_δ aquí). Este es el enfoque elegido para introducir restricciones en este trabajo.

2.3. Planteamiento del problema

En el marco de optimización de procesos con restricciones, donde existe un coste de experimentación y un presupuesto limitado a N experimentos, el problema a resolver en este trabajo es: Con un modelo probabilístico definido por (1)-(3), y dados los históricos $\mathcal{B}_f$ y $\mathcal{B}_g$, encontrar la secuencia de decisiones futuras $X_N = \{x_1, x_2, \dots, x_N\}$ que optimiza

$$J \coloneqq \mathbb{E}\left\{\gamma^N J_N(Y_{f,N}) + \sum_{k=1}^{N-1} \gamma^k\, r(Y_{f,k})\right\} \tag{6}$$

sobre el conjunto de N observaciones futuras (denotadas por $Y_{f,N} \coloneqq \{y_f(x_1), \dots, y_f(x_N)\}$ y $Y_{g,N} \coloneqq \{y_g(x_1), \dots, y_g(x_N)\}$) tal que $G_\delta(\mathcal{B}_g, x_k) < 0\ \forall k$, es decir, cumpliendo con las restricciones estimadas del proceso con probabilidad $1-\delta$.

En la expresión (6), $r(\cdot)$ es una función de coste inmediato y $J_N(\cdot)$ una función de coste terminal, a decidir dependiendo de la aplicación concreta. Nótese que (6) incluye el objetivo de la OB estándar como caso particular, tomando $r(Y_{f,k}) = 0$ y $J_N(Y_{f,N}) = \min Y_{f,N}$ como función de coste terminal*. De igual forma, también incluye otros casos de interés en la optimización de procesos, como por ejemplo el caso que considera el coste o beneficio de cada experimento mediante $r(Y_k) = y_f(x_k)$ y $J_N(Y_N) = \min Y_{f,N}$, véase (Armesto et al., 2023) para otras opciones de potencial interés. El factor de descuento γ se utiliza para guiar el proceso de optimización, modulando si se desea asegurar resultados aceptables desde el principio ($\gamma > 1$) o permitir una mayor exploración inicial para tener mejores resultados hacia el final del horizonte ($\gamma < 1$).

Las heurísticas de OB no tienen en consideración el coste de experimentación ni el límite de experimentos de forma explícita. Son, por tanto, *miopes* en cuanto a la toma de decisión se refiere. Además, según el problema a optimizar, una heurística puede ofrecer mejores resultados que otra, pero esto a priori es complicado de inferir. Es por ello que la OB estándar no es la solución adecuada al problema planteado en este trabajo. Nótese que (6) es en realidad un problema POMDP, porque las decisiones sobre x_k dependerán no sólo del conjunto de datos históricos, sino también de las observaciones futuras inciertas $y_k, \dots, y_N$ (por tratarse de un problema MDP sobre PGs, es decir, sobre el conjunto de datos disponibles). En la siguiente sección se presenta la solución propuesta basada en un árbol de escenarios y programación dinámica.

3. Estrategia OB-POMDP

La combinación de un modelo probabilístico, que aparece debido a la incertidumbre existente en las predicciones, y la programación dinámica requiere considerar un árbol de escenarios con las posibles acciones y observaciones resultantes. Dicho árbol se construye a partir de un conocimiento inicial, cargando los dos procesos gaussianos $(\mathcal{B}_f, \mathcal{B}_g)$ con el histórico de observaciones experimentales previas. Cada escenario evalúa un conjunto de decisiones $\tilde{x}_l \in \mathcal{X}$ y observaciones "virtuales" de f, denotado $\tilde{y}_l \in \mathcal{Y}$, para $l: 1, \dots, N-k+1$. Es decir, estima el desempeño de las decisiones que quedan por tomar hasta agotar el presupuesto u horizonte N. La decisión seleccionada para la primera etapa, $\tilde{x}_1$, será aquella presuntamente factible que obtenga el menor coste esperado. Ésta será por tanto la candidata x_k que testar experimentalmente, y sus respuestas medidas $y_f(x_k)$, $y_g(x_k)$ servirán para actualizar el conocimiento probabilístico adquirido sobre el desempeño real f y las restricciones g.

Como el problema POMDP con espacio de decisiones y observaciones continuo es intratable (Paulson et al., 2022), hay que aproximar su solución limitando la expansión del árbol. En nuestra propuesta, la creación de escenarios se restringirá a evaluar un conjunto de decisiones prometedoras (e.g., las proporcionadas por heurísticas de OB) por medio de posibles observaciones obtenidas a partir de la cuadratura de Gauss/Hermite. La decisión óptima del nodo raíz será la candidata a aplicar experimentalmente, en el contexto de OB. A continuación, detallamos la estructura de dicho árbol y en la Sección 3.3 describimos el algoritmo OB-POMDP propuesto en su conjunto.

3.1 Árbol de escenarios

El árbol de escenarios mencionado se construirá con dos tipos de nodos: **nodos V** y **nodos Q**. Los nodos Q son hijos de los nodos V. De igual manera, los nodos V son a su vez hijos de los nodos Q, a excepción del nodo raíz que es del tipo V. El conjunto $\mathcal{C}_\mathcal{N}$ denotará los $\mathcal{N}'$ hijos de un cierto nodo denotado por $\mathcal{N}$, mientras que $\mathcal{P}_\mathcal{N}$ denota el nodo padre de $\mathcal{N}$. Los nodos V están asociados a decisiones, mientras que los nodos Q son necesarios para promediar observaciones (asociados a cada decisión del nodo padre de tipo V). La Figura 1 ilustra de forma esquemática el árbol de escenarios descrito, que pasaremos a esbozar conceptualmente a continuación. El pseudocódigo completo de su funcionalidad se adjunta en el apéndice al final del artículo, para el lector interesado en el detalle de implementación.

Como es habitual en programación dinámica, cada nodo tendrá asociado un estado, que en un POMDP es el conocimiento probabilístico de f, aquí representado por $\mathcal{B}_{\mathcal{N},f}$, y una

* En OB se busca la decisión que mejor desempeño obtiene sobre la función estática f, sin tener en cuenta directamente el nº de experimentos necesarios.

función de valor. Si un nodo, denotado por $\mathcal{N}$, es de tipo V, entonces estima la función de valor

$$V_{\mathcal{N}} \coloneqq \min_{\mathcal{N}' \in \mathcal{C}_{\mathcal{N}}} Q_{\mathcal{N}'},$$

recursivamente a partir de la de sus hijos*, de tipo Q, generados por ramificación con las decisiones candidatas $\tilde{x}_i \in \mathcal{X}$. Dichas decisiones candidatas son calculadas mediante optimización:

$$\tilde{x}_{\mathcal{N},i} \coloneqq \arg\min \text{ o } \max_{\substack{x \in \mathbb{X} \\ G_\delta(\mathcal{B}_g, x) < 0}} F_i(\mathcal{B}_{\mathcal{N},f}, x) \tag{7}$$

Siendo $F_i \in \mathcal{F}$ una función de adquisición perteneciente a una lista de ellas $\mathcal{F}$ decidida a priori.

La función de valor $Q_{\mathcal{N}}$ de un nodo tipo Q se estima mediante la cuadratura de Gauss-Hermite (Abramowitz y Stegun, 1972). En efecto, para limitar la complejidad del árbol, en lugar de muestrear extensivamente el PG de f para aproximar la función de valor, utilizaremos un conjunto de M valores $\tilde{y}_j$ (observaciones virtuales), con sus pesos asociados α_j, dados por la cuadratura de Gauss-Hermite. El valor esperado de $Q_{\mathcal{N}}$ se aproxima, por tanto, con los estimados asociados a cada y_j, análogo a la idea que subyace en el desarrollo del filtro *unscented* de Kalman (Wan y van der Merwe, 2001). La fórmula final es también recursiva, ya que depende de la función de valor de los nodos hijos de tipo V, denotados como $\mathcal{N}' \in \mathcal{C}_{\mathcal{N}}$:

$$Q_{\mathcal{N}} \coloneqq \sum_{j=1}^{M} \alpha_j \left(r(\tilde{y}_j \cup Y_{\mathcal{P}_{\mathcal{N}}}) + \gamma V_{\mathcal{N}_j'} \right)$$

Todas estas ideas forman la base conceptual de dos métodos asociados a las clases que implementan cada nodo:

1) EXPANDIR, para construir el árbol de escenarios de forma recursiva, creando otros nodos del tipo que corresponda por ramificación; y

2) FUNCIONVALOR, para evaluar cada escenario y estimar la función de valor asociada a cada nodo por propagación inversa.

Además, para evitar un excesivo coste de cómputo, el árbol se expande únicamente hasta alcanzar una profundidad H_b decidida por el usuario. A partir de esa profundidad, se debe calcular un valor estimado del coste hasta el final del horizonte que, en general, dependerá de $r(\cdot)$ y $J_N(\cdot)$. Este valor estimado se calcula con la función COSTEFINALESPERADO (detalles en el pseudocódigo incluido en el apéndice).

3.2 Opciones de exploración ampliada

Nótese que, en (7), el PG que modela las restricciones es independiente del nodo, porque se actualiza sólo con observaciones reales de las restricciones. Por tanto, resolver (7) para un límite de confianza superior fijo implica asegurar factibilidad en la predicción con probabilidad $1-\delta$, pero puede dar lugar a un comportamiento excesivamente conservador, porque se mantiene el mismo espacio de búsqueda durante todas las etapas de predicción, lo que impide explorar decisiones futuras que probablemente podrían cumplir las restricciones.

Para paliar esta limitación sin incrementar drásticamente el coste computacional de la exploración POMDP, en este trabajo planteamos relajar los límites de confianza superior que definen $G_\delta(\mathcal{B}_g, x)$ en (7) a medida que se profundiza en la evaluación del árbol de decisión, incrementando el valor de δ (i.e. disminuyendo η como en la Figura 1). También ampliamos el conjunto de funciones de adquisición $\mathcal{F}$ con dos heurísticas propias de carácter exploratorio:

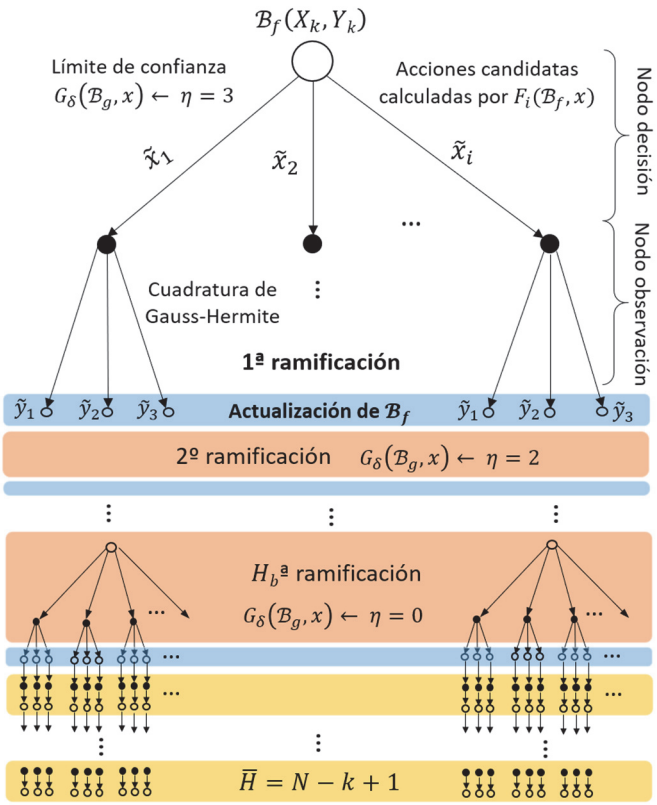

Figura 1: Árbol de escenarios generado por la estrategia propuesta.

- Exploración de restricción (CE, *Constraint Exploration*): Selecciona la decisión x factible según $G_\delta(\mathcal{B}_g, x) < 0$ cuya predicción marginal es de varianza máxima, esto es, explora el extremo "seguro" de la restricción que presenta mayor incertidumbre a priori en $\mathcal{B}_g$.
- Exploración local (LE, *Local Exploration*): Elige la siguiente decisión entre los vértices de un hipercubo de tamaño predefinido (e.g., la mitad del rango admisible de x) centrado en la última decisión tomada en $\mathcal{B}_{\mathcal{N},f}$. El vértice elegido será el más alejado del centroide calculado con las decisiones pasadas que cumpla $G_\delta(\mathcal{B}_g, x) < 0$. Si ningún vértice cumple restricciones o todos ya fueron explorados previamente, se selecciona un punto factible al azar dentro del hipercubo planteado.

3.3 Algoritmo maestro OB-POMDP

El objetivo del algoritmo desarrollado es decidir cuál será la siguiente decisión x_k, $k: 1, \ldots, N$, a experimentar dado el histórico de datos experimentales $\mathcal{B}_f$ y $\mathcal{B}_g$, recolectados hasta $k-1$, y sus PGs asociados.

El pseudocódigo del Algoritmo 1 calcula una decisión x dado el conocimiento probabilístico actual sobre f y g, la ejecuta experimentalmente, actualiza dicho conocimiento con la observación realizada, y repite el procedimiento hasta que el presupuesto de N decisiones se ha agotado.

* Cuando un nodo V no tiene hijos, su función de valor es directamente el coste terminal.

Algoritmo 1. OB-POMDP

Global: $X, Y_f, Y_g, N, \bar{H}$

1: Crear $\mathcal{B}_f\{X, Y_f\}$ y $\mathcal{B}_g\{X, Y_g\}$ % Crear los PG iniciales
2: **Para** $k = 1$ **hasta** N **hacer:**
3: $b = \text{NODOV}(\mathcal{B}_f)$ % Crear el nodo V raíz
4: $\bar{H} \leftarrow N - k + 1$ % Reducir horizonte de predicción
5: $b.\text{EXPANDIR}()$ % Ramificar el árbol
6: $\{V, x\} \leftarrow b.\text{FUNCIONVALOR}()$ % Estimar la función de valor por propagación inversa
7: Aplicar la decisión x % Realizar experimento real
8: $y_f \leftarrow f(x); y_g \leftarrow g(x)$ % Observar la respuesta
9: $\mathcal{B}_f \leftarrow \mathcal{B}_f \oplus \{x, y_f\}; \mathcal{B}_g \leftarrow \mathcal{B}_g \oplus \{x, y_g\}$ % Actualizar PG

El algoritmo comienza creando PGs con la información disponible a priori del sistema. Con este conocimiento inicial y la lista de funciones de adquisición $\mathcal{F}$, la clase NODOV crea el nodo de decisión raíz. Tal y como se ha mencionado anteriormente, los nodos creados tienen implementados los métodos: EXPANDIR y FUNCIONVALOR. Este último debe proporcionar tanto el valor esperado del coste J a futuro, como la decisión óptima asociada que, en el caso del nodo raíz, será la que se realice experimentalmente.

Paralelización. Evidentemente, la eficiencia computacional del Algoritmo 1 mejora drásticamente si se paraleliza la evaluación del árbol de decisión. La forma más obvia de hacerlo es paralelizar la creación de nodos en el método EXPANDIR, según las medidas virtuales dadas por la cuadratura GH y la lista de funciones de adquisición $\mathcal{F}$. Esto daría lugar a la creación de hilos de procesamiento anidados, que puede o no ser un inconveniente para implementar una paralelización efectiva dependiendo del lenguaje de programación utilizado. En este trabajo se utilizó MATLAB® para la implementación, que no permite anidar bucles ***parfor***. En consecuencia, hemos diseñado los métodos EXPANDIR y FUNCIONVALOR para salvar esta dificultad. Por claridad en la exposición, los detalles y el pseudocódigo correspondiente se adjuntan en el apéndice.

3.4 Estudio comparativo sin restricciones

Para evaluar el desempeño esperado de un algoritmo de optimización según la minimización de un índice de coste (6) es necesario generar un conjunto de funciones de test amplio. Para ello en este trabajo hemos promediado el desempeño de los algoritmos testados en 250 realizaciones (i.e. funciones f a optimizar) de un PG de dos variables de decisión en el cuadrado de vértices ± 1, de media cero, con función exponencial cuadrática como generador de covarianza

$$\kappa(x_a, x_b) = \Sigma \cdot e^{-\frac{\|x_b - x_a\|^2}{2 \cdot \sigma^2}}$$

y considerando un ruido blanco de medida definido por $\lambda = 0.002$. La elección de esta función y sus parámetros ha sido arbitraria, si bien, consideramos que esta elección no es relevante porque es común a todos los algoritmos evaluados.

El presupuesto máximo ha sido fijado en $N = 7$, porque en ese horizonte las estrategias de OB estándar consiguen una estimación razonable del óptimo. En esta evaluación no se consideró la inclusión de restricciones de proceso g por simplicidad. Hemos realizado la comparativa ante dos conjuntos diferentes de hiperparámetros del PG generador (filas en la Tabla 1). Además de las 4 opciones de OB miope listadas en la Sección 2.1, también se han comparado los siguientes algoritmos de la literatura relacionados con la presente propuesta:

- *GP-HEDGE*. Algoritmo miope que selecciona la siguiente acción a tomar de entre la lista de heurísticas de OB proporcionadas, basándose en la probabilidad de obtener mejor ganancia/recompensa, calculada a partir del acumulado de ganancias potencialmente obtenidas por cada heurística en muestras pasadas (Hoffman et al., 2011).
- *2-OPT*. OB anticipada a dos pasos. Para ser justos en la comparativa con el índice de coste (6), esta función de adquisición ha sido modificada para optimizar EV a horizonte N en lugar del EI propuesto en Wu y Frazier (2019).
- *ROLL*. Algoritmo de OB anticipada a H_b pasos con optimización anidada según Lam et al. (2016). Al igual que en el caso anterior, hemos modificado la política por defecto EI propuesta en Lam et al. (2016) por la heurística EV para ser coherentes con nuestro índice (6).

Los resultados de la Tabla 1 han sido obtenidos seleccionando $H_b = 3$ tanto para ROLL como para nuestro algoritmo OB-POMDP*. Como era razonable esperar, el algoritmo ROLL que selecciona la primera decisión en el espacio continuo $\mathbb{X}$ resolviendo una optimización anidada, obtiene el menor coste medio acumulado en ambos test. No obstante, nuestra propuesta POMDP, que selecciona la primera decisión sin optimización anidada sólo estimando cual es la heurística de OB más prometedora, es la siguiente que mejor coste medio obtiene: sólo un 2.5% peor que ROLL, pero requiriendo menos de la mitad de carga computacional (12.25s versus 5.42s de media en el banco de pruebas evaluado).

La función de adquisición de OB con anticipación a dos pasos 2-OPT es un caso particular de ROLL fijando $H_b = 2$, por lo que era esperable que su desempeño fuera ligeramente peor. Sin embargo, el coste también es un 5.5% peor que el obtenido por el nuestro, a pesar de resolver una optimización anidada. Nótese que las necesidades computacionales de los algoritmos ROLL y 2-OPT van a escalar peor que las del POMDP con el nº de variables y la adicción de restricciones, ya que no requiere resolver una optimización anidada.

El algoritmo GP-HEDGE comparte con nuestra propuesta la selección de la siguiente decisión entre la cartera de heurísticas disponible (en este caso, las de la Sección 2.1), pero su coste medio observado en los test es claramente peor, comparable al obtenido por las opciones de OB estándar en la clase de problemas que proponemos abordar en este artículo.

Tabla 1: Desempeño medio observado con los algoritmos evaluados en 2 bancos de pruebas consistentes en 250 funciones generadas aleatoriamente.

σ	Σ	EI	PI	LCB	EV
0.5	1	7.45	7.24	8.5	8.1
0.35	2	14.6	13.4	15.6	13.6

σ	Σ	ROLL	POMDP	2-OPT	HEDGE
0.5	1	**5.9**	**6.05**	6.38	7.9
0.35	2	**12.1**	**12.4**	13.1	13.7
Uso CPU:		12.25s	5.42s	2.04s	0.086s

* Por cuestiones de espacio en las figuras y tablas, el Algoritmo 1 será denotado simplemente por POMDP cuando no exista posibilidad de confusión.

4. Aplicación a un proceso de producción por lotes: Reactor de Otto-Williams con restricciones

A continuación, se evalúa la eficacia del algoritmo propuesto en un caso de estudio que emula un proceso industrial.

4.1. Descripción del proceso a optimizar

El conocido modelo propuesto por Williams y Otto en 1960 es un reactor de tanque continuamente agitado (CSTR por sus siglas en inglés) donde existen dos entradas de material reactivo: F_A que no es controlable pero sí conocido, y F_B que es variable de decisión. La temperatura del reactor T_R es la otra variable de decisión que se puede manipular para obtener una determinada calidad del producto a la salida al final del lote.

En el interior del reactor ocurren tres reacciones químicas en paralelo, existiendo en total 6 componentes: 4 productos (C, E, G y P) y las partes de A y B que quedan por reaccionar.

$$\begin{aligned} A + B &\overset{k_1}{\rightarrow} C;\ \ k_1 = 1.66 \cdot 10^6 \cdot e^{-\frac{6666.7}{273+T_R}} \\ B + C &\overset{k_2}{\rightarrow} P + E;\ \ k_2 = 7.2177 \cdot 10^8 \cdot e^{-\frac{8333.3}{273+T_R}} \\ C + P &\overset{k_3}{\rightarrow} G;\ \ k_3 = 2.6745 \cdot 10^{12} \cdot e^{-\frac{11111}{273+T_R}} \end{aligned} \quad (8)$$

Las ecuaciones que describen este proceso pueden consultarse, por ejemplo, en el trabajo Zhang y Forbes (2000).

Tabla 2: Precios, parámetros del proceso y límites de operación.

Param.	Valor	Unidad	Param.	Valor	Unidad
C_A	76.23	€/kg	F_B^{UP}	3	kg/s
C_B	114.34	€/kg	F_B^{LO}	6	kg/s
P_P	1143.38	€/kg	T_R^{UP}	100	ºC
P_E	25.92	€/kg	T_R^{LO}	70	ºC
F_A	1.8275	kg/s	V_R	2105	kg

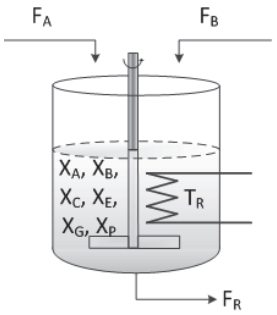

Figura 2: Diagrama simplificado del reactor de Otto-Williams.

El objetivo planteado en este trabajo es optimizar la receta de producción (i.e., decidir el valor de F_B y T_R dentro de unos límites de operación) de lote a lote, de forma que se obtenga el máximo beneficio económico acumulado al producir $N = 8$ lotes de los productos presentes en el caudal de salida F_R:

$$J \coloneqq \sum_{k=1}^{6} \gamma^k \left(\left(X_{P,k} \mathrm{P_P} + X_{E,k} \mathrm{P_E} \right) F_{R,k} - F_A \mathrm{C_A} - F_{B,k} \mathrm{C_B} \right) \quad (9)$$

En (9), $\mathrm{P_i}$ son los precios de venta de los productos y $\mathrm{C_j}$ los costes de los reactivos. Se decidió un factor de descuento $\gamma = 0.98$ para favorecer la producción de lotes de buena calidad ya desde el principio. Nótese que el beneficio (9) está definido directamente sobre caudales, por lo que el coste total necesitaría obtenerse multiplicando por la duración del lote (no incluido en la fórmula porque no influye para nada en el resultado de la optimización).

No obstante, debe respetarse la siguiente restricción sobre la máxima concentración de producto G presente en cada lote:

$$X_{G,k} < 9.5\% \quad k\colon 1, \dots, 6 \quad (10)$$

4.2. Configuración del algoritmo OB-POMDP y resultados

El Algoritmo 1 ha sido configurado de la siguiente manera. El conocimiento probabilístico acerca del reactor consiste en dos PGs. El PG asociado a f ha sido definido por

$$\kappa_f(x_a, x_b) \coloneqq \bar{\kappa}_f(x_a, x_b) + \phi_f(x) \Sigma_{\theta_f} \phi_f(x)^T$$

con $\bar{\kappa}_f(x_a, x_b) = \Sigma_{\bar{\kappa}_f} \cdot e^{-\frac{\|x_b - x_a\|^2}{2 \cdot \sigma_f^2}}$, $\Sigma_{\bar{\kappa}_f} = 64$, $\sigma_f = \sqrt{0.15}$, $\Sigma_{\theta_f} = 1000$, $\phi_f(x) = 1$ y $\lambda_f = 0.5$.

Los valores mostrados han sido elegidos de tal forma que estiman una desviación estándar en el beneficio económico por lote de 8€ alrededor de su valor medio, y una correlación de 0.3 al desplazarse un 30% sobre el rango de operación*. Estos valores se podrían deducir en base al conocimiento que se tenga sobre este tipo de procesos, teniendo en cuenta el beneficio registrado de producir lotes de producto similar en el pasado (datos históricos), y de las hojas de características de los analizadores de concentración y caudalímetros.

Los hiperparámetros del PG asociado a g también se han elegido a priori, con un procedimiento de inferencia análogo†:

$$\kappa_g(x_a, x_b) \coloneqq \bar{\kappa}_g(x_a, x_b) + \phi_g(x) \Sigma_{\theta_g} \phi_g(x)^T$$

con $\bar{\kappa}_g(x_a, x_b) = \Sigma_{\bar{\kappa}_g} \cdot e^{-\frac{\|x_b - x_a\|^2}{2 \cdot \sigma_g^2}}$, $\Sigma_{\bar{\kappa}_g} = 5$, $\sigma_g = \sqrt{0.15}$, $\Sigma_{\theta_g} = 100$, $\phi_g(x) = 1$ y $\lambda_g = 0.1$.

Como es usual para dar prioridad a la estimación del parámetro de media constante o "bias" θ_g, su varianza Σ_θ ha sido fijada a un valor suficientemente grande en comparación con la parte no paramétrica.

El algoritmo se iniciaría desde el mejor punto de operación que se conociera por el histórico de la planta. En este ejemplo sólo se asume la existencia de un único punto de operación para iniciar los algoritmos

$$x_0 \coloneqq \{F_B = 5.6 \mathrm{kg\,s^{-1}}, T_R = 81 \mathrm{°C}\}$$

cuyo beneficio es $y_0 = 130.38$ €/lote. Sin embargo, el óptimo del proceso sin violar la restricción (10) se encuentra en

$$x^* \coloneqq \{F_B = 4.89 \mathrm{kg\,s^{-1}}, T_R = 87.64 \mathrm{°C}\}$$

que corresponde con un beneficio (sin considerar ruido de medida) de $y^* = 188.92$ €/lote.

A continuación, se ha comparado el desempeño del Algoritmo 1 con el que consigue la OB estándar con cada una de las 4 funciones de adquisición listadas en la Sección 2.1. También se compara con nuestra adaptación de los algoritmos GP-HEDGE, 2-OPT, y el ROLL de Lam et al., (2017), para que incluyan restricciones duras según el criterio de la Sección 2.2. Tanto nuestro algoritmo POMDP como el ROLL se ejecutaron

* Las variables de decisión x han sido normalizadas al rango [-1,1].

† Nótese que no se asume la disponibilidad de un conjunto de datos iniciales representativos para realizar una estimación mediante máxima verosimilitud.

con horizonte de ramificación $H_b = 3$. Los resultados se resumen en los números de la Tabla 3 y la Figura 3.

Tabla 3: Pérdida de beneficio (€) acumulada respecto a producir 8 lotes óptimos sin existencia de ruido de medida.

EI	PI	LCB	EV	ROLL	2-OPT	HEDGE	POMDP
122.5	173.1	108.2	135.6	102.8	96.85	91.45	83.46

Todos los algoritmos han partido del mismo conocimiento inicial acerca del proceso (un único dato x_0, y_o) y ninguno ha tenido acceso al modelo mecanístico del reactor (utilizado como caja negra actuando de "planta experimental").

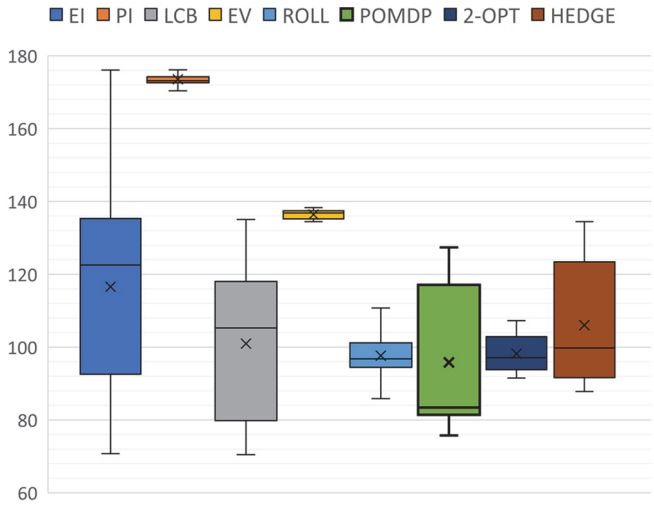

Figura 3: Estadística de la pérdida de beneficio (€) acumulada respecto a producir 8 lotes óptimos en 18 ensayos independientes de los algoritmos. La mejor mediana es la observada para nuestro algoritmo POMDP.

La Tabla 3 muestra el desempeño (calculado en pérdida de beneficio acumulada respecto a haber conocido el óptimo inicialmente) de cada algoritmo al producir 8 lotes con un reactor de sensores perfectos (i.e. sin ruido de medida w). No obstante, los PGs utilizados en los algoritmos no son conscientes de esto. Los resultados obtenidos en estas condiciones ideales aúpan a nuestro Algoritmo 1 como el que mejor desempeño obtiene, seguido por el GP-HEDGE. Esto refuerza la premisa sobre la utilidad de elegir una función de adquisición de forma dinámica entre experimentos, frente a resolver una más costosa optimización anidada. De hecho, ni la OB estándar con LCB (la mejor de las heurísticas clásicas probadas) obtiene un desempeño comparable al del Algoritmo 1 en este test.

Sin embargo, la existencia de sensores perfectos no es realista. Por ello se añadió un ruido aleatorio no despreciable en las medidas que influye en la toma de decisiones en los primeros lotes. Con el fin de presentar una comparativa más justa en el caso realista, hemos ejecutado todos los algoritmos en 18 ensayos independientes de 8 lotes cada uno, partiendo todos desde el mismo punto x_0, pero con realizaciones aleatorias del ruido de medida. Los datos de desempeño obtenidos se resumen en el diagrama de cajas y bigotes de la Figura 3.

La primera conclusión que puede extraerse es que la existencia de ruido de medida influye sustancialmente en este caso de estudio (se observan importantes variaciones de desempeño acumulado para un mismo algoritmo) donde el conocimiento inicial del sistema es ciertamente escaso.

Aunque no se pueden extraer conclusiones estadísticamente fuertes con pocas ejecuciones, nuestra propuesta OB-POMDP es la que mejor desempeño medio ha obtenido, y claramente la que tiene una probabilidad más elevada de obtener lotes de producto con mayores beneficios.

Los algoritmos ROLL y 2-OPT obtuvieron resultados más consistentes entre ejecuciones, i.e., demostraron ser más robustos al ruido que las heurísticas EI y LCB de OB clásica. Sin embargo, el coste computacional extra que implican no se ha traducido en un mejor desempeño medio, incluso en comparación con OB-LCB.

Los datos parecen corroborar que la estrategia de selección dinámica de la heurística de OB más adecuada es la que predice hacia adelante en el tiempo estimando la función de valor por programación dinámica aproximada, en lugar de calcular la probabilidad de potencial ganancia basándose en las ganancias estimadas a posteriori que realiza GP-HEDGE.

Por ejemplo, para los resultados de la Tabla 3, la secuencia de decisiones tomadas por el algoritmo OB-POMDP fue:

1. Límite de confianza inferior (LCB)
2. Exploración de restricción (CE)
3. Valor esperado (EV)
4. Valor esperado (EV)
5. Valor esperado (EV)
6. Valor esperado (EV)
7. Probabilidad de mejorar (PI)
8. Valor esperado (EV)

Mientras que las tomadas por el algoritmo GP-HEDGE fueron:

1. Valor esperado (EV)
2. Límite de confianza inferior (LCB)
3. Límite de confianza inferior (LCB)
4. Límite de confianza inferior (LCB)
5. Límite de confianza inferior (LCB)
6. Límite de confianza inferior (LCB)
7. Límite de confianza inferior (LCB)
8. Límite de confianza inferior (LCB)

Véase como nuestro algoritmo OB-POMDP decide usar heurísticas de carácter exploratorio al inicio, en lugar de ir al valor esperado (EV) como decide el GP-HEDGE. Este comportamiento es más coherente con la intuición de que primero puede valer la pena explorar para adquirir cierto conocimiento sobre el proceso para seguidamente explotarlo, en lugar de explotar y luego explorar para tratar de mejorar.

A modo ilustrativo, las figuras 4 y 5 muestran la secuencia de experimentos realizados por nuestro algoritmo OB-POMDP correspondientes a la ejecución sin ruido de medida: la primera sobre el beneficio por lote y la restricción real del reactor, y la segunda sobre el modelo de conocimiento probabilístico adquirido después de realizar dichos experimentos. En la Figura 4 puede apreciarse como en ningún caso se violó la restricción, pese a que con una información del proceso tan escasa a priori y utilizando modelos probabilísticos no existen garantías totales de cumplimiento a la hora de ejecutar el experimento real. Finalmente, la Figura 6 muestra las secuencias de experimentos realizadas sobre las curvas de nivel reales del reactor por aquellos algoritmos que obtuvieron los mejores desempeños en la prueba con sensores sin rudo de medida.

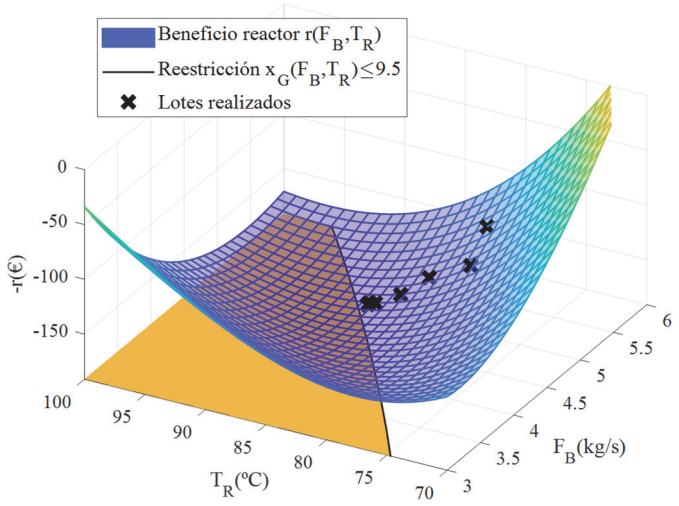

Figura 4: Desempeño real del reactor y experimentos ejecutados por el algoritmo OB-POMDP partiendo de $x_0 = [5.6\ \text{kg s}^{-1}, 81°\text{C}]$.

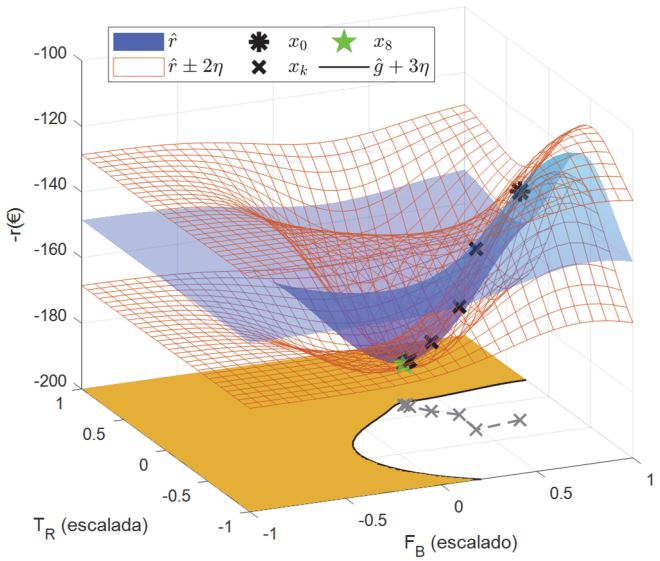

Figura 5: Conocimiento probabilístico construido después de ejecutar los 6 lotes por el algoritmo OB-POMDP partiendo de x_0. La zona segura está delimitada por el límite de confianza superior (UCB) de la restricción estimado con $\eta = 3$ veces la desviación típica marginal del PG.

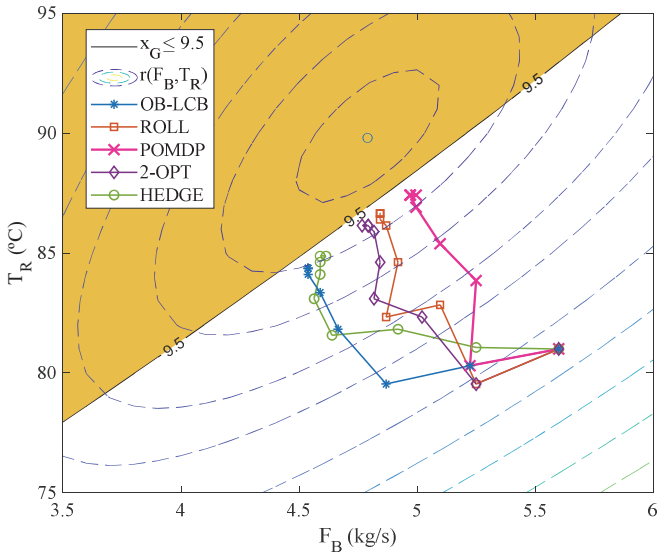

Figura 6: Secuencias de decisiones tomadas por los algoritmos con mejor desempeño observado partiendo desde x_0 sin medidas contaminadas por ruido.

Es interesante observar como nuestro OB-POMDP toma como primera decisión la proporcionada por OB-LCB, pero luego una heurística de exploración pura (teóricamente más subóptima) proporciona un conocimiento valioso que, a la larga, resulta en un mayor beneficio acumulado. Los algoritmos ROLL, 2-OPT, y OB-POMDP son los que más se aproximan al óptimo real en el lote final, a pesar de que el GP-HEDGE obtuvo un beneficio acumulado mejor que el obtenido por ROLL y 2-OPT. A la vista de los desempeños obtenidos y de la similitud de las decisiones tomadas por los algoritmos ROLL con $H_b = 3$ y 2-OPT (equivale a ROLL con $H_b = 2$), podría deducirse que en este problema no existe una ganancia adicional significativa que justifique el coste computacional extra que implica ramificar a horizontes mayores.

5. Conclusiones

La toma óptima de decisiones en casos donde existe un limitado presupuesto para experimentación requiere plantear el problema como un POMDP para tratar correctamente el dilema de exploración–explotación. El algoritmo OB-POMDP planteado aquí ha demostrado que poder decidir la función de adquisición más prometedora a lo largo del horizonte de predicción mejora a las heurísticas miopes de OB, particularmente en el problema en el que se tiene un coste inmediato asociado a cada experimento realizado, por lo que se penaliza no sólo el valor de coste final obtenido, si no también todo el acumulado.

El algoritmo implementado para aproximar el problema POMDP planteado tiene en consideración heurísticas bien reconocidas de la OB, agrupa ramas del árbol de decisión cuando las funciones de adquisición proporcionan resultados similares, utiliza una discretización eficiente de las observaciones más probables gracias a cuadratura de Gauss-Hermite, e incluye aspectos de paralelización para acelerar la evaluación del árbol. Todo esto ha demostrado su eficacia, tanto en un amplio banco de pruebas con funciones sintéticas, como en un problema académico que aproxima un caso de estudio inspirado en la industria química y de procesos, donde se trata de optimizar un beneficio acumulado de forma segura, i.e., sin violar restricciones de calidad (lo que conllevaría desechar el lote producido).

Está claro que la búsqueda de decisiones en el espacio continuo a la hora de formular el POMDP es lo óptimo conceptualmente. Sin embargo, los resultados obtenidos comparando con la propuesta de Lam et al. (2017) prueban que nos podemos aproximar bastante limitándonos a evaluar unas pocas decisiones dadas por unas buenas heurísticas en etapas iniciales. Esto permite un sustancial ahorro en tiempo de cómputo, que podría aprovecharse para ramificar a más profundidad y aproximar mejor la función de valor de cada decisión candidata.

En general, todas las estrategias de OB sufren de una alta dependencia del ruido de medida e hiperparámetros del PG; nótese que las afirmaciones sobre el desempeño de este tipo de algoritmos se ofrecen en términos estadísticos (valor esperado). Tener esta idea clara es importante en casos de procesos reales donde sólo hay una única oportunidad para tomar decisiones y evaluar el desempeño (como el ejemplo de producir unos pocos lotes con el reactor químico).

Finalmente, aunque nuestra premisa era un planteamiento de optimización puramente basado en datos, en la práctica, la disponibilidad de un modelo mecanístico aproximado del proceso es clave para el éxito de algoritmos basados en OB.

Agradecimientos

Esta investigación se ha desarrollado en el marco del proyecto LOCPU (PID2020-116585GB-I00) financiado por MCIN/AEI/10.13039/501100011033.

Referencias

Abramowitz, M., Stegun, I.A., 1972. Handbook of mathematical functions, 10th printing with corrections, Dover Publications, ISBN: 978-0-486-61272-0. [Ecuación 25.4.46]

Armesto, L., Pitarch, J.L., Sala, A., 2023. Acquisition function choice in Bayesian optimization via partially observable Markov decision process. IFAC-PapersOnLine, 56(2), 1572-1577.

Armesto, L., Sala, A., 2022. Volume-weighted Bellman error method for adaptive meshing in approximate dynamic programming. Revista Iberoamericana de Automática e Informática industrial, 19(1), 37–47. DOI: 10.4995/riai.2021.15698

Astudillo, R., Jiang, D., Balandat, M., Bakshy, E., Frazier, P., 2021. Multi-step budgeted Bayesian optimization with unknown evaluation costs. Advances in Neural Information Processing Systems, 34, 20197-20209.

Busoniu, L., Babuska, R., De Schutter, B., Ernst, D., 2017. Reinforcement learning and dynamic programming using function approximators. CRC press. DOI: 10.1201/9781439821091

Calandra, R., Seyfarth, A., Peters, J., Deisenroth, M.P., 2016. Bayesian optimization for learning gaits under uncertainty. Annals of Mathematics and Artificial Intelligence 76, 5–23. DOI: 10.1007/s10472-015-9463-9

Cressie, N., 1990. The origins of kriging. Mathematical Geolology 22, 239–252. DOI:10.1007/BF00889887

Deisenroth, M.P., Neumann, G., Peters, J., 2013. A survey on policy search for robotics. Foundations and Trends® in Robotics 2, 1–142. DOI:10.1561/2300000021

del Rio Chanona, E.A., Petsagkourakis, P., Bradford, E., Graciano, J.E.A., Chachuat, B., 2021. Real-time optimization meets Bayesian optimization and derivative-free optimization: A tale of modifier adaptation. Computers & Chemical Engineering 147, 107249. DOI: 10.1016/j.compchemeng.2021.107249

Duvigneau, R., Chandrashekar, P., 2012. Kriging-based optimization applied to flow control. International Journal for Numerical Methods in Fluids, 69(11), 1701-1714. DOI: 10.1002/fld.2657

Frazier, P.I., 2018. Bayesian optimization, in: Recent advances in optimization and modeling of contemporary problems. Informs, 255–278. DOI: 10.1287/educ.2018.0188

Gardner, J.R., Kusner, M.J., Xu, Z., Weinberger, K.Q., Cunningham, J.P., 2014. Bayesian optimization with inequality constraints. Proceedings of the 31st Inter. Conf. on Machine Learning, ICML, 937-945.

Gelbart, M.A., Snoek, J., Adams, R.P., 2014. Bayesian optimization with unknown Constraints. Proceedings of the 30th Conf. on Uncertainty in Artificial Intelligence, UAI, 250-259.

Girbés-Juan, V., Moll, J., Sala, A., Armesto, L., 2023. Cautious Bayesian optimization: A line tracker case study. Sensors 23(16), 7266. DOI: 10.3390/s23167266

Hoffman, M., Brochu, E., De Freitas, N., 2011. Portfolio allocation for Bayesian optimization. UAI, pp. 327-336.

Lam, R., Willcox, K., Wolpert, D.H., 2016. Bayesian optimization with a finite budget: An approximate dynamic programming approach. Advances in Neural Information Processing Systems 29, 883-891.

Lam, R., Willcox, K., 2017. Lookahead Bayesian optimization with inequality constraints. Advances in neural information processing systems 30.

Letham, B., Karrer, B., Ottoni, G., Bakshy, E., 2019. Constrained Bayesian optimization with noisy experiments. Bayesian Analysis 14(2), 495-519. DOI: 10.1214/18-BA1110

Mora, J.P., Samper, J., Carlos F., 2023. Estudio de la optimización Bayesiana para reducir el consumo energético de un robot paralelo durante tareas pick and place. Revista Iberoamericana de Automática e Informática industrial, 20(1), pp. 1–12. DOI: 10.4995/riai.2022.16724

Paulson, J.A., Sorouifar, F., Chakrabarty, A., 2022. Efficient multi-step lookahead Bayesian optimization with local search constraints. IEEE 61st Conference on Decision and Control (CDC), 123-129. DOI: 10.1109/CDC51059.2022.9992943

Jaiswal, P., Honnappa, H., Rao, V.A., 2023. Bayesian joint chance constrained optimization: Approximations and statistical consistency. SIAM Journal on Optimization, 33(3), 1968-1995. DOI: 10.1137/21M1430005

Pitarch, J.L., Armesto, L., Sala, A., Montes, D., 2023. Optimización experimental con presupuesto finito combinando heurísticas Bayesianas en un POMDP. XLIV Jornadas de Automática, 447-452. DOI:10.17979/spudc.9788497498609.447

Rasmussen, C. E., Williams, C. K., 2006. Gaussian processes for machine learning. Cambridge, MA: MIT press.

Rodríguez-Blanco, T., Sarabia, D., Pitarch, J.L., de Prada, C., 2017. Modifier adaptation methodology based on transient and static measurements for RTO to cope with structural uncertainty. Computers & Chemical Engineering 106, 480–500. DOI: 10.1016/j.compchemeng.2017.07.001

Spaan, M.T.J., 2012. Partially observable Markov decision processes. (eds. Wiering, M., van Otterlo, M.) Reinforcement Learning. Springer, 387–414. DOI: 10.1007/978-3-642-27645-3_12

Wan, E.A., van der Merwe, R., 2001. The unscented Kalman filter. Kalman Filtering and Neural Networks (ed. Haykin S.), 221-280. DOI: 10.1002/0471221546.ch7

Wu, J., Frazier, P., 2019. Practical two-step lookahead Bayesian optimization. Advances in neural information processing systems, 32.

Wu, T., Movellan, J., 2012. Semi-parametric Gaussian process for robot system identification. IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 725-731. DOI: 10.1109/IROS.2012.6385977

Yip, W.S., Marlin, T.E., 2003. Designing plant experiments for real time optimization systems. Control Engineering Practice 11, 837–845. Process Dynamics and Control. DOI: 10.1016/S0967-0661(02)00213-7

Zhang, Y., Forbes, J.F., 2000. Extended design cost: A performance criterion for real-time optimization systems. Computers & Chemical Engineering 24(8), 1829-1841. DOI: 10.1016/S0098-1354(00)00561-5

Apéndice A. Pseudocódigo de la implementación

Este apéndice presenta una versión de pseudocódigo para los tipos de nodos V y Q representado por las clases NODOV y NODOQ. Ambas usan propiedades y métodos clásicamente usados en estructuras arbóreas de programación, como son: $\mathcal{C}$, que representa la lista de nodos "hijos"; $\mathcal{P}$,el nodo "padre"; y el método NODOSHOJA, que devuelve una lista de nodos sucesores que son "hojas", es decir, que no tienen hijos.

Se asume que las clases tienen acceso a una serie de elementos globales según se requieran, que forman el conjunto de parámetros y funciones necesarias para el algoritmo, y que han sido detalladas previamente. Concretamente, son elementos globales: la función de coste inmediato r; la función de coste terminal J_N; el factor de descuento γ; la profundidad de expansión del árbol H_b; las funciones de media y el conjunto de datos históricos $\mathcal{B}_g$; la lista $\mathcal{F}$ con las funciones de adquisición a usar durante el proceso; y la lista $\mathcal{G}$ con los límites de confianza superior de g expresando diferentes niveles de relajación a medida que se profundiza en el árbol (aumenta δ conforme el árbol avanza). Además, se asumen disponibles dos funciones que determinan la ramificación del árbol. Por un lado, la función GHQ_M devuelve un punto de la cuadratura Gauss-Hermite (la tolerancia Δ_y determina el número de puntos M con los que queremos estimar el valor de un nodo del tipo Q) y, por otro lado, la función uniqueTol devuelve acciones distanciadas al menos con una cierta tolerancia Δ_x, para evitar duplicar innecesariamente ramas del árbol con decisiones muy parecidas entre ellas, que acabarán produciendo resultados similares.

Nótese que el método EXPANDIR sólo ramifica como máximo hasta una profundidad H_b dada. Desde H_b en adelante los nodos tienen un único hijo, hasta la profundidad máxima del árbol $\bar{H}$ (Figura 1), lo que permite estimar el valor de cada nodo a profundidad H_b mediante la función COSTEFINALESPERADO. Dicha función en general implicará una optimización determinista que depende de $\mathcal{B}_{\mathcal{N},f}$, $\mathcal{B}_g$, r y J_N.

No obstante, cuando r y J_N dependen exclusivamente de la observación actual, como en (9), se puede evitar resolver optimizaciones innecesarias. En ese caso, la decisión $\tilde{x}^*$ que minimiza (o maximiza) el valor esperado $\mu_{\mathcal{B}_{\mathcal{N},f}}(x)$, sujeto a $G_\delta(\mathcal{B}_g, x) < 0$, se mantiene activa hasta final del horizonte.

Pseudocódigo Nodos V

Global: $\mathcal{F}, \mathcal{G}, \Delta_x, H_b, \bar{H}, \mathcal{B}_g$

Clase NODOV$(\mathcal{B}_f)$
  Propiedades Almacenar $\mathcal{B}_f$, $V_N \leftarrow 0$
  Método EXPANDIR()
    Para $i = 1$ **hasta** $\min(H_b, \bar{H})$ **hacer:**
      $nodosV \leftarrow$ NODOSHOJA()
      Para cada $\mathcal{N}$ **en** $nodosV$ **hacer:**
        Para $j = 1$ **hasta** longitud$(\mathcal{F})$ **hacer:**
          $\mathcal{N}.\mathcal{C}[j] \leftarrow$ NODOQ$(\mathcal{B}_f, \mathcal{F}[j], \mathcal{G}[i])$
        $nodosQ \leftarrow$ NODOSHOJA()
        Para cada $\mathcal{N}$ **en** $nodosQ$ **hacer en paralelo:**
          $\mathcal{N}.\tilde{x} \leftarrow \arg\min_{x|\mathcal{N}.G(\mathcal{B}_g,x)<0} \mathcal{N}.F(\mathcal{B}_f, x)$
            % Optimiza las funciones de adquisición
        Para cada $\mathcal{N}$ **en** $nodosV$ **hacer:**
          Para $j = 1$ **hasta** longitud$(\mathcal{F})$ **hacer:**
            $\tilde{X}[j] \leftarrow \mathcal{N}.\mathcal{C}[j].\tilde{x}$
          $[\tilde{X}, I] \leftarrow$ uniqueTol$(\tilde{X}, \mathcal{N}.\mathcal{B}_f.X, \Delta_x)$
          $\mathcal{N}.\mathcal{C} \leftarrow \mathcal{N}.\mathcal{C}[I]$ % Poda acciones similares
          Para cada $\mathcal{N}'$ **en** $\mathcal{N}.\mathcal{C}$ **hacer:**
            $\mathcal{N}'$.EXPANDIR()
  Método FUNCIONVALOR()
    Si $\mathcal{C} = \emptyset$ **entonces:**
      Devuelve V_N,nulo
    En otro caso:
      Si $\mathcal{P} = \emptyset$ **entonces:**
        $nodosV \leftarrow$ NODOSHOJA()
        Para cada $\mathcal{N}$ **en** $nodosV()$ **hacer en paralelo:**
          $V_N \leftarrow$ COSTEFINALESPERADO$(\mathcal{N})$
        Para $i = 1$ **hasta** longitud$(\mathcal{C})$ **hacer:**
          $Q[i] \leftarrow \mathcal{C}[i]$.FUNCIONVALOR()
        $\{V, idx\} \leftarrow \min Q$
        Devuelve $V, \mathcal{C}[idx].\tilde{x}$

La clase NODOV ha sido diseñada para paralelizar la ejecución de las funciones de adquisición $F_i(\mathcal{B}_f, x)$ por niveles de profundidad del árbol, en lugar de la creación anidada de hilos a la que daría lugar la expansión del árbol de forma recursiva. Más concretamente, las líneas 10 y 11 de la clase NODOV permiten ejecutar en paralelo optimizaciones de los nodos terminales (hasta esa profundidad, según va creciendo el árbol). De igual forma, las líneas 25 y 26 permiten calcular en paralelo el coste final esperado cuando el árbol se ha expandido hasta profundidad H_b.

Pseudocódigo Coste Final Esperado

Global: $r, \gamma, J_N, \Delta_x, H_b, \bar{H}, \mathcal{B}_g$

Función COSTEFINALESPERADO$(\mathcal{N})$
  $\tilde{x}^* \leftarrow \arg\min_{x|\mathcal{N}.G(\mathcal{B}_g,x)\leq 0} \mu_{\mathcal{N}.\mathcal{B}_f}(x)$
  $V \leftarrow 0$
  Para $i = H_b + 1$ **hasta** $\bar{H} - 2$ **hacer:**
    $V \leftarrow V + \gamma^{i-1} r\left(\mu_{\mathcal{N}.\mathcal{B}_f}(\tilde{x}^*)\right)$ %Repite la mejor acción
  $V \leftarrow V + \gamma^{\bar{H}-1} J_N\left(\mu_{\mathcal{N}.\mathcal{B}_f}(\tilde{x}^*)\right)$
  Devuelve V

Pseudocódigo Nodos Q

Global: r, γ, Δ_y

Clase NODOQ$(\mathcal{B}_f, F, G)$
  Propiedades Almacenar $\mathcal{B}_f, F, G$; Declarar $\alpha, \tilde{x}$
  Método EXPANDIR() % Ramificación no recursiva
    Calcular $\mu_{\mathcal{B}_f}(\tilde{x})$ y $\kappa_{\mathcal{B}_f}(\tilde{x}, \tilde{x})$ a partir de (1) y (2)
    $\kappa_{\mathcal{B}_f}(\tilde{x}, \tilde{x}) < \Delta_y ? M \leftarrow 1 : M \leftarrow 3$
      % Cuadratura GH adaptativa
    Para $j = 1$ **hasta** M **hacer:**
      $\{\tilde{y}_f, \alpha[j]\} \leftarrow$ GHQM$(\mu_{\mathcal{B}_f}(\tilde{x}), \kappa_{\mathcal{B}_f}(\tilde{x}, \tilde{x}), j)$
        % Punto de cuadratura GH
      $\mathcal{C}[j] \leftarrow$ NODOV$(\mathcal{B}_f \oplus \{\tilde{x}, \tilde{y}_f\})$ %Actualizar PG de f
  Método FUNCIONVALOR()
    Para $j = 1$ **hasta** longitud$(\mathcal{C})$ **hacer:**
      $\mathcal{N}' \leftarrow \mathcal{C}[j]$
      $V \leftarrow \mathcal{N}'$.FUNCIONVALOR()
      $Q[j] \leftarrow \left(r(\mathcal{B}_f.Y) + \gamma V\right)$
  Devuelve $\sum_j \alpha[j] \cdot Q[j]$