ScienceDirect
Disponible en www.sciencedirect.com
Revista Iberoamericana de Automática e Informática industrial 12 (2015) 385–396
www.elsevier.es/RIAI

Control Multimodal en Entornos Inciertos usando Aprendizaje por Refuerzos y Procesos Gaussianos

Mariano De Paula[a,*], Luis O. Ávila[b], Carlos Sánchez Reinoso[c], Gerardo G. Acosta[a].

[a]*Núcleo INTELYMEC – CIFICEN – CONICET, Facultad de Ingeniería, Universidad Nacional del Centro de la Provincia de Buenos Aires – UNCPBA, Av. del Valle 5737, Olavarría B7400JWI, Argentina.*
[b]*Instituto de Desarrollo y Diseño- INGAR (CONICET-UTN), Avellaneda 3657, Santa Fe S3002 GJC, Argentina.*
[c]*Centro de Diseño y Optimización de Sistemas, Facultad de Tecnología y Ciencias Aplicadas, Universidad Nacional de Catamarca-CONICET, Maximio Victoria 55, San Fernando del Valle de Catamarca 4700 SFV.*

Resumen

El control de sistemas complejos puede ser realizado descomponiendo la tarea de control en una secuencia de modos de control, o simplemente modos. Cada modo implementa una ley de retroalimentación hasta que se activa una condición de terminación, en respuesta a la ocurrencia de un evento exógeno/endógeno que indica que la ejecución del modo debe finalizar. En este trabajo se presenta una propuesta novedosa para encontrar una política de conmutación óptima para resolver el problema de control optimizando alguna medida de costo/beneficio. Una política óptima implementa un programa de control multimodal óptimo, el cual consiste en un encadenamiento de modos de control. La propuesta realizada incluye el desarrollo y formulación de un algoritmo basado en la idea de la programación dinámica integrando procesos Gaussianos y aprendizaje Bayesiano activo. Mediante el enfoque propuesto es posible realizar un uso eficiente de los datos para mejorar la exploración de las soluciones sobre espacios de estados continuos. Un caso de estudio representativo es abordado para demostrar el desempeño del algoritmo propuesto.

Palabras Clave:
Control multimodal, Programación dinámica, Procesos Gaussianos, Incertidumbre, Política.

1. Introducción

La creciente complejidad de los sistemas dinámicos y la incertidumbre acerca de las condiciones operativas hacen que el diseño de las estrategias de control se vuelva cada vez más complejo. En un enfoque de control multimodal se busca resolver una tarea de control compleja a través del encadenamiento de modos de control, o simplemente modos (Mehta y Egerstedt 2008). Los sistemas así controlados pueden pensarse como sistemas dinámicos híbridos con una trayectoria en tiempo continuo que puede ser representada como una secuencia de evoluciones continuas intercaladas por eventos discretos (Lunze y Lehmann 2010). Cada una de estas evoluciones depende del modo de control que es aplicado durante el intervalo de tiempo correspondiente. Esto da lugar a un *muestreo de Lebesgue* (Åström y Bernhardsson 2003) en el que los intervalos de muestreo no son constantes y/o no siguen un patrón periódico. Este tipo de muestreo tiene especial relevancia en situaciones donde la complejidad de control tiene que ser ponderada en función del tiempo de ejecución (Azuma, Imura, y Sugie 2010). Para los sistemas así operados, el problema de control óptimo con muestreo de Lebesgue se ha formulado escasamente y, más aún, hay muy pocos antecedentes que incorporen condiciones de incertidumbre en la formulación de tales problemas, probablemente debido a la dificultad de analizar este tipo de situaciones (Xu y Cao 2011).

En este trabajo se propone un algoritmo para aprendizaje capaz de encontrar una política de conmutación para la selección de modos, basándose en los principios de la programación dinámica (PD) (Bertsekas 2000) usando simulaciones, que se combina con muestreo de Lebesgue y aproximaciones con procesos Gaussianos (GP) (Rasmussen y Williams 2006). Para hacer frente al tamaño y dimensión de espacio de estados (EE) y a un continuo de parámetros de la ley de retroalimentación, el aprendizaje Bayesiano activo (Deisenroth, Rasmussen, y Peters 2009) se incorpora mediante el empleo de una función de utilidad que permite balancear la información contenida con la mejora de la política de conmutación. Sobre la base de los datos obtenidos por una influencia creciente de las condiciones de operación, se obtienen modelos de GPs mediante aprendizaje. Estos modelos representan la dinámica de transición de estados, de acuerdo a la ejecución de cada modo(Deisenroth 2010). También se emplean modelos de GPs para describir las funciones de valor, empleadas por los algoritmos de aprendizaje, en dominios continuos.

La estructura de este artículo es la siguiente. La Sección 2 describe algunos trabajos relacionados con el control multimodal. En la Sección 3 se describe la evolución de los sistemas

*Autor en correspondencia.
Correos electrónicos: mariano.depaula@fio.unicen.edu.ar
URL: www.fio.unicen.edu.ar

http://dx.doi.org/10.1016/j.riai.2015.09.004

dinámicos manipulados mediante una estrategia de control multimodal. Además, se introduce la metodología de aprendizaje por refuerzos para control multimodal y se muestra un caso de estudio que es utilizado para poner a prueba cada uno de los algoritmos expuestos en las diferentes etapas del presente trabajo, consistente en el guiado de una embarcación, que se desplaza con velocidad constante, desde un punto de partida hasta un punto de llegada, en un río con un cierto perfil de corriente de agua. En la Sección 4 se presenta una propuesta algorítmica de programación dinámica que emplea muestreo de Lebesgue, aplicada sobre el caso de estudio. En la Sección 5 se formula una propuesta de programación dinámica con muestreo de Lebesgue que incorpora GP y aprendizaje activo, la cual es apta para encontrar una política de control multimodal para manipular un sistema dinámico bajo incertidumbre. La Sección 6 expone las conclusiones obtenidas a partir de esta investigación.

2. Trabajos relacionados

Para un número finito de modos de operación, recientemente se planteó un novedoso paradigma de modelado conocido como autómata híbrido en tiempo continuo, denominado en inglés como *integral continuous-time hybrid automata* (icHA). El mismo propone un control de eventos basado en la solución de un problema de optimización donde no se asume información a priori acerca del momento y orden de los eventos (Di Cairano, Bemporad, y Júlvez 2009). Según la PD, el control óptimo de un icHA queda implícitamente determinado por las ecuaciones asociadas con las condiciones de Hamilton-Jacobi-Bellman (HJB) y la solución de control óptimo debe ser obtenida mediante una combinación eficiente del muestreo del espacio de estados y la aproximación de la función de valor. En el trabajo de Zhang y Colab. (2009), la PD aproximada ha sido exitosamente aplicada a problemas de control en tiempo discreto para sistemas regulados con lógicas de conmutación en tiempo discreto.

Desde la perspectiva del control óptimo no lineal, cuando la PD es aplicada a sistemas en tiempo continuo con ocurrencia de eventos discretos el problema computacional resultante es costoso de resolver dado que usualmente abarca programas matemáticos no convexos (Xu y Antsaklis 2003). Muchos trabajos previos, relacionados con la aplicación del formalismo de PD al control óptimo y planificación de secuencia de modos para sistemas con lógicas de conmutación, revelan la cuestión de los problemas de optimización combinatoria con complejidad exponencial (Borrelli et al. 2005; Lincoln y Rantzer 2006; Rantzer 2006).

La solución de los problemas de optimización dinámica con autómatas híbridos incorporados y en tiempo continuo (Bemporad y Morari 1999), ha sido profusamente revisada por Barton y Colab. (2006). En esta última contribución los autores proponen un enfoque para dividir al problema original en otros problemas más simples. Sobre esta base se propone una metodología de búsqueda con cortes y limitaciones por aproximaciones exteriores heurísticas para resolver los problemas de optimización no convexos mixtos-enteros formulados. Aún cuando la importante problemática de la incertidumbre para la optimización de sistemas dinámicos híbridos no ha sido abarcada en el trabajo de Barton y Colab. (2006), su importancia ha sido reconocida principalmente en relación con la verificación de seguridad de un sistema dinámico donde interactúan variables discretas y continuas.

Desde la perspectiva del control basado en el comportamiento, tomado del campo de la robótica, Mehta y Edgerstedt (2006) también trataron el importante problema de identificar secuencias óptimas de modos en el control óptimo multimodal usando técnicas de aprendizaje por refuerzos (Sutton y Barto 1998; Deisenroth, Rasmussen, y Peters 2009; Busoniu et al. 2010; Pajares Martin-Sanz y De la Cruz Garcia 2010). El aprendizaje por refuerzos es una técnica conveniente para resolver problemas de control óptimo bajo incertidumbre y demuestra ser efectiva para abordar el problema de concatenar o combinar modos de control con leyes de retroalimentación y condiciones de terminación pre-establecidas. Para sobrellevar el inconveniente de un número fijo de modos, Mehta y Egerstedt proponen aumentar el conjunto de modos como una función de los modos actuales usando cálculo variacional (Mehta y Egerstedt 2008). Además, para un programa de control dado (secuencia de modos), Mehta y Egerstedt formularon un problema de control óptimo para mejorar el desempeño del sistema a través de la adición de nuevas leyes de control. Esto lo lograron mediante una combinación óptima de concatenaciones recurrentes de modos. La incertidumbre en los estados iniciales es el mayor obstáculo para el control multimodal dado que los programas de control se derivan asumiendo condiciones iniciales fijas. Debido a ésto es que el enfoque de control multimodal propuesto resulta inefectivo para hacer frente a las perturbaciones endógenas y eventos inciertos que afecten la evolución del sistema en busca de su objetivo final.

Para capturar completamente la dinámica de un sistema híbrido controlado donde los estados continuos interactúan con el estado discreto en un entorno incierto, si bien la literatura existente no es profusa, existen contribuciones relevantes que merecen ser mencionadas (Bensoussan y Menaldi 2000; Shi et al. 2006; Cassandras y Lygeros 2007; Abate et al. 2008; Adamek, Sobotka, y Stursberg 2008; Blackmore et al. 2010; Song y Li 2010; Bemporad y Di Cairano 2011). El trabajo de Bensoussan y Menaldi (Bensoussan y Menaldi 2000) es un aporte muy valioso sobre el uso de la programación dinámica para el control estocástico de sistemas híbridos, mientras que en el libro de Cassandras y Lygeros (2007) se resalta la importancia de la incertidumbre en el control óptimo de sistemas híbridos. Para abordar problemas de control óptimo computacionalmente intratables, para una clase de sistemas híbridos estocásticos con ecuaciones lineales de estados y una función de costo separable, Song y Li (Song y Li 2010) propusieron recientemente un esquema de control cuasi óptimo de retroalimentación basado en métodos estadísticos de predicción. Juntamente a las suposiciones de linealidad, otra desventaja de los métodos numéricos aproximados de solución es que realizan un muestreo exhaustivo e innecesario sobre todo el espacio de estados.

En Blackmore et al. (2010) se desarrolla una aproximación de control usando la metáfora de las partículas la cual está restringida a sistemas lineales o dinámicas Markovianas lineales con discontinuidades y carece de una estrategia orientada a un objetivo para definir la duración de las etapas de decisión. Por otra parte, la solución propuesta debe ser recalculada en tiempo real para diferentes condiciones iniciales. En el enfoque propuesto por Blackmore y Colab., en lugar de un enfoque de programación dinámica, el problema aproximado es resuelto usando técnicas de programación lineal mixta-entera.

La solución de PD fue aplicada para verificación probabilística de condiciones terminales y validación de seguridad en el control de sistemas estocásticos híbridos en tiempo discreto (Abate et al. 2008). Se determinan políticas markovianas de "máxima confiabilidad" para obtener el conjunto de condiciones iniciales, dando una cierta garantía probabilística de que el sistema

evolucionará dentro de una región segura del espacio de estado, el cual es caracterizado en términos de una función de valor.

Bemporad y Di Cairano (Bemporad y Di Cairano 2011) se focalizan en el control óptimo con horizonte deslizante para sistemas con lógicas de conmutación lineales, usando autómatas estocásticos discretos híbridos (*Discrete Hybrid Stochastic Automata* – DHSA) en los cuales las transiciones de estados discretas dependen de la ocurrencia tanto de eventos determinísticos como estocásticos. Se usa un enfoque de control óptimo en tiempo finito para determinar la trayectoria que brinda el mejor balance entre el desempeño en el seguimiento de la trayectoria y la probabilidad de que ésta se ejecute bajo restricciones probabilísticas. Las principales desventajas para el uso del paradigma DHSA son los modelos lineales, un número finitos de modos y la falta de una política de control del tipo *orientada a una meta* (*goal-directed*).

Finalmente, es importante señalar la estrategia de control de "modo deslizante" para sistemas con discontinuidades estocásticas propuesta por Shi y Colab. (2006), en el cual los parámetros de cada "salto" son modelados como un proceso de Markov homogéneo en tiempo continuo con estados discretos. El modo describe las discontinuidades estocásticas de los parámetros del sistema y la ocurrencia de discontinuidades.

3. Aprendizaje de programas de control multimodal

3.1. Dinámica multimodal

Cada modo implementa una ley de control de retroalimentación hasta que, por alguna razón, se interrumpe su ejecución. El momento en que ocurre la interrupción de la ejecución de un determinado modo está asociado con la activación de una condición de terminación del modo ξ_q. Esta condición se activa en respuesta a un determinado evento/ perturbación, como puede ser el logro de una meta, el alcance de un determinado estado del sistema, la violación de una restricción o la ocurrencia de algún evento externo al sistema (Liberzon 2003; Egerstedt, Wardi, y Axelsson 2006; Axelsson et al. 2007; Ding, Wardi, y Egerstedt 2009).

Formalmente, cada modo de control $\mathbf{\Sigma}_i$ queda definido por una ley de control de realimentación $\mathcal{K}_i(\mathbf{x})$ y condiciones de terminación $\boldsymbol{\xi}_i(\mathbf{x},\mathbf{d})$ que determinarán el fin de la ejecución del modo que esté activo, siendo $\mathbf{x}$ el estado del sistema y $\mathbf{d}$ las perturbaciones. $\boldsymbol{\xi}_i$ se define como un vector binario el cual al activarse cambia su valor de 0 a 1, indicando que la ejecución del *i-ésimo* modo activo debe ser detenido. Para cada modo, la dinámica en tiempo continuo generalmente es modelada por ecuaciones diferenciales que describen la evolución del estado del sistema.

En un esquema de control multimodal, asumiendo que se tiene a priori una determinada secuencia finita de modos de control $\Xi = [(\mathcal{K}_1,\boldsymbol{\xi}_1),\ldots,(\mathcal{K}_N,\boldsymbol{\xi}_N)]$ para operar un determinado sistema, la evolución del mismo resultará:

$$\begin{aligned}\dot{\mathbf{x}} &= f\left(\mathbf{x},(\mathcal{K}_1,\boldsymbol{\xi}_1)\right), t_0 < t < \tau_1, \mathbf{x}(t_0)=\mathbf{x}_0, \mathbf{d}(t_0)=\mathbf{d}_0\\ &\vdots\\ \dot{\mathbf{x}} &= f\left(\mathbf{x},(\mathcal{K}_N,\boldsymbol{\xi}_N)\right), \tau_{N-1} \le t < \tau_N, \mathbf{x}(\tau_{N-1})=\mathbf{x}_{N-1}\end{aligned} \quad (1)$$

dónde la acción de control aplicada, durante la ejecución de un *i-ésimo* modo, es $\mathbf{u}(t)=\mathcal{K}_i(\mathbf{x})\ \forall\ \tau_{i-1}\le t<\tau_i$, tal que $t=\tau_i$ es el instante de conmutación cuando $\boldsymbol{\xi}_i=1$. Esta forma de operar un sistema da lugar a una secuencia de evoluciones continuas intercaladas con la ocurrencia de eventos discretos.

3.2. Aprendizaje de un programa multimodal

En una aplicación de control multimodal existe la necesidad de influir el comportamiento de los sistemas para alcanzar una determinada meta u objetivo. Así, resolver un problema de este tipo puede traducirse en hallar una política de actuación (Salichs et al. 2010) que permita obtener un desempeño óptimo del sistema de acuerdo a la maximización, o minimización, de alguna medida de rendimiento (ganancia o costo).

A través de la resolución de un problema de aprendizaje por refuerzos es posible aprender iterativamente una política de conmutación de modos para el controlador a partir de las interacciones con el sistema o entorno con el fin de alcanzar una determinada meta u objetivo. En el marco del aprendizaje por refuerzos (RL), en sus trabajos Mehta y Egerstedt han sido pioneros en propuestas para control multimodal de sistemas (Mehta y Egerstedt 2005; Mehta y Egerstedt 2006). Para resolver un problema de (RL) utilizan el clásico algoritmo Q-*Learning* tabular, proponiendo una forma ingeniosa para discretizar el espacio de estados basándose en el muestreo de Lebesgue.

Dado un conjunto finito de modos de control $\mathcal{M}=\{\mathbf{\Sigma}_1,\mathbf{\Sigma}_2,\ldots,\mathbf{\Sigma}_M\}$, un controlador selecciona y ejecuta una secuencia posible de modos $\Xi=\mathbf{\Sigma}_i, i=1\ldots N$, con vista a maximizar la recompensa recibida desde un par estado/perturbación inicial $(\mathbf{x}_0,\mathbf{d}_0)$ hasta que se alcanza un determinado estado objetivo. Bajo una política de conmutación dada π, asúmase que la recompensa acumulada esperada $V^{\pi}(\mathbf{x}_0,\mathbf{d}_0)$ para un determinado intervalo de tiempo es una función de $(\mathbf{x}^{\pi},\mathbf{d},\mathbf{t}^{\pi},\mathbf{\Sigma}^{\pi})$, donde $\mathbf{t}=\{t(k)\}_{k=1}^{k=N}$ es un vector cuyos elementos son los instantes de tiempo en los cuales ocurren los cambios de modos, $\mathbf{x}^{\pi}=\{\mathbf{x}(k)\}_{k=1}^{k=N}$ son los estados correspondientes en esos instantes de tiempo, $\mathbf{d}=\{\mathbf{d}(k)\}_{k=1}^{k=N}$ son las perturbaciones en tales momentos y $\Xi^{\pi}=\{\mathbf{\Sigma}(k)\}_{k=1}^{k=N}$ define la secuencia de modos de control de acuerdo a la política de conmutación π. La secuencia $\mathbf{x}^{\pi}$ de transiciones de estados da lugar a las recompensas $\{r(k)\}_{k=1}^{k=N}$. Así, esta secuencia de recompensas puede ser usada para definir la función de valor de estado como (Sutton y Barto 1998):

$$V^{\pi}(\mathbf{x}_0,\mathbf{d}_0) \rightarrow E[\gamma^N r(N)+\sum_{k=1}^{N-1}\gamma^i r(k)] \quad (2)$$

dónde $\gamma\in[0,1]$ es el factor de descuento que establece el valor presente de las recompensas futuras. De esta forma, una política de conmutación óptima π^* será aquella que maximice la función de valor asociada en (2), tal que la secuencia generada $(\mathbf{x}^{\pi},\mathbf{d},\mathbf{t}^{\pi},\mathbf{\Sigma}^{\pi})$ con su correspondiente secuencia de recompensas $\{r(k)\}_{k=1}^{k=N}$ satisfacen la ecuación de valor recursiva de Bellman:

$$\begin{aligned}V^*(\mathbf{x}(k),\mathbf{d}(k)) = \arg\max_{\Sigma\in\mathcal{M}}\{r(k+1)+\\ \gamma\left(V^*(\mathbf{x}(k+1))|\mathbf{x}(k),\mathbf{d}(k),\mathbf{\Sigma}(k)\right).\end{aligned} \quad (3)$$

Similarmente, la función de valor de estado-acción Q* puede definirse como:

$$\begin{aligned}Q^*(\mathbf{x}(k),\mathbf{d}(k),\mathbf{\Sigma}(k)) = r(k+1)+\\ \gamma\max_{\Sigma'\in\mathcal{M}}\left(V^*(\mathbf{x}(k+1))|\mathbf{x}(k),\mathbf{d}(k),\mathbf{\Sigma}'(k)\right)\end{aligned} \quad (4)$$

tal que $V^*(\mathbf{x},\mathbf{d})=\arg\max_{\mathbf{\Sigma}} Q^*(\mathbf{x},\mathbf{d},\mathbf{\Sigma}')$ para todo par $(\mathbf{x},\mathbf{d})$. Una vez que Q* es aprendida a través de las interacciones simuladas, la política de conmutación se puede obtener directamente mediante:

$$\pi^*(\mathbf{x},\mathbf{d}) = \arg\max_{\Sigma\in\mathcal{M}} Q^*(\mathbf{x},\mathbf{d},\mathbf{\Sigma}). \quad (5)$$

Algoritmo 1. Control multimodal (Mehta & Edgersted)

1: **Input**: $\alpha, \gamma, \mathcal{M}$
2: $\mathcal{X} := \{\mathbf{x}_0, f(\mathbf{x}_0, (\mathcal{K}, \boldsymbol{\xi})\}\ \forall\ (\mathcal{K}, \boldsymbol{\xi}) \in \mathcal{M}$
3: $step(\mathbf{x}_0) := 0$
4: $p := 1$
5: $step(f(\mathbf{x}_0, (\mathcal{K}, \boldsymbol{\xi})) := 1\ \forall\ (\mathcal{K}, \boldsymbol{\xi}) \in \mathcal{M}$
6: $Q(\mathbf{x}, (\mathcal{K}, \boldsymbol{\xi})) := const\ \forall\ \mathbf{x} \in \mathcal{X}$
7: **Repeat**
8: $p := p + 1$
9: $\tilde{\mathbf{x}} := rand(\mathbf{x} \in \mathcal{X})$
10: $(\mathcal{K}, \boldsymbol{\xi}) := rand(\mathcal{M})$
11: $\tilde{\mathbf{x}}' := f(\tilde{\mathbf{x}}, (\mathcal{K}, \boldsymbol{\xi}))$
12: **If** $\tilde{\mathbf{x}}' \notin \mathcal{X}$ **Then**
13: $step(\tilde{\mathbf{x}}') := step(\tilde{\mathbf{x}}) + 1$
14: $\mathcal{X} := \mathcal{X} \cup \{\tilde{\mathbf{x}}'\}$
15: $Q(\tilde{\mathbf{x}}', (\mathcal{K}, \boldsymbol{\xi})) = const\ \forall\ (\mathcal{K}, \boldsymbol{\xi}) \in \mathcal{M}$
16: **End If**
17: $Q(\tilde{\mathbf{x}}, (\mathcal{K}, \boldsymbol{\xi})) = Q(\tilde{\mathbf{x}}, (\mathcal{K}, \boldsymbol{\xi})) + \alpha \left(r(\tilde{\mathbf{x}}, (\mathcal{K}, \boldsymbol{\xi})) + \gamma \max_{(\mathcal{K}, \boldsymbol{\xi})} Q(\tilde{\mathbf{x}}', (\mathcal{K}, \boldsymbol{\xi})) - Q(\tilde{\mathbf{x}}, (\mathcal{K}, \boldsymbol{\xi}))\right)$
18: **Until** $mod(p, L) = 0\ and\ \left| Q_p(\tilde{\mathbf{x}}, (\mathcal{K}, \boldsymbol{\xi})) - Q_{p-L}(\tilde{\mathbf{x}}, (\mathcal{K}, \boldsymbol{\xi}))\right| < \epsilon,\ \forall\ \mathbf{x} \in \mathcal{X},\ (\mathcal{K}, \boldsymbol{\xi}) \in \mathcal{M}$

Tabla 1. Algoritmo de control multimodal.

En la Tabla 1 se describe una versión de un algoritmo de control multimodal para encontrar una política de conmutación óptima π^* a partir de la función de valor de estado-acción óptima Q^* (Mehta y Egerstedt 2005). Como se puede observar, este algoritmo es una versión sencilla para resolver un problema de RL. El mismo requiere como entrada los parámetros de entrenamiento α y γ; y además debe definirse el conjunto de modos $\mathcal{M}$, en el que cada modo queda especificado mediante su ley de realimentación y condiciones de terminación. La función $step(\tilde{\mathbf{x}})$ representa el menor número de modos encadenados para alcanzar el estado $\tilde{\mathbf{x}}$ desde $\mathbf{x}_0$. L representa un número suficientemente grande de repeticiones.

3.3. Ejemplo de un barco dirigido a una meta

Para poner a prueba y demostrar las bondades de los algoritmos desarrollados a lo largo de este trabajo, vamos a recurrir a un ejemplo estudiado por Bryson y Ho (1975) en el marco del control óptimo. El mismo es un ejemplo simplificado. Consiste en guiar un barco, que se desplaza con velocidad constante, desde un punto de partida hasta un punto final en un río con un cierto perfil de corriente de agua. La Fig. 1 esquematiza los distintos componentes del problema. El objetivo del problema consiste en encontrar una trayectoria que insuma el menor tiempo posible. La dinámica del sistema queda definida por el siguiente conjunto de ecuaciones diferenciales (Rosenstein y Barto 2004):

$$\dot{x} = C\cos(\phi) - y \cdot C \qquad (6)$$
$$\dot{y} = C\sin(\phi)$$

donde el término $y.C$ representa el efecto perturbante de la corriente de agua en el desplazamiento de la embarcación. En este problema, el barco debe partir desde la posición inicial $\mathbf{x}_0 = (-3.66, -1.86)^T$ para alcanzar la región objetivo de $0.1\ km$ de radio centrada en $\mathbf{x}_{goal} = (0,0)^T$. El barco se desplaza a velocidad constante $C = 0.01$ km/s. Se asume que cada vez que se aplica una determinada acción de control (rumbo del timón, ϕ) se debe mantener durante un período de 25 segundos (período de muestreo). Cuando ocurre un cambio en la dirección del timón, se asume que tal cambio ocurre de forma instantánea. La trayectoria óptima encontrada por Bryson y Ho (1975) insume un total de 536.6 segundos. Esta solución fue obtenida para un escenario totalmente determinista. La motivación del trabajo que se presenta en este artículo es, justamente, relajar estas suposiciones a una situación más parecida a la realidad de operación de este tipo de embarcaciones, contemplando un ambiente estocástico. De este modo, este ejemplo debe ser visto simplemente como un punto de partida para experimentar con nuevas ideas.

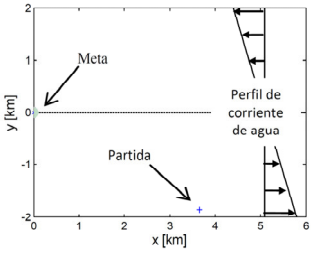
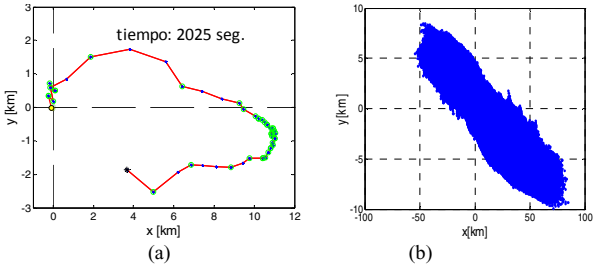


Figura1. Ejemplo de barco dirigido a una meta

Para una aplicación del Algoritmo 1 de la Tabla 1 se emplean valores de $\alpha = 0.1$ y $\gamma = 0.95$. Como función de recompensa $r(\cdot)$, se considera un valor de -1 siempre que se arribe a un estado que no pertenece a la región objetivo y un valor de +10 cuando se alcanza esa región. Se define un conjunto $\mathcal{M}$ de 36 modos, para los cuales la ley de realimentación de cada uno de los modos $\boldsymbol{\Sigma}_i \in \mathcal{M}, i = 0 \ldots 35$ corresponde con un valor fijo ϕ, lo que representa la acción aplicada sobre el sistema. Las condiciones de terminación son idénticas para todos los modos, esto es $\boldsymbol{\xi}_i = \boldsymbol{\xi}\ \forall\ \boldsymbol{\Sigma}_i \in \mathcal{M}$. Formalmente, cada modo se define mediante la ley de control: $\mathcal{K}_i(\mathbf{x}) = \phi_i\ \forall\ \phi_i = i \cdot 10°, i = 0,1, \ldots, 35$ y el parámetro binario $\boldsymbol{\xi} = \xi_a \vee \xi_b \vee \xi_c$ tal que $\boldsymbol{\xi}_i = 0 \Leftrightarrow \xi_a = \xi_b = \xi_c = 0$; de lo contrario $\boldsymbol{\xi} = 1$. Para todos los modos las funciones lógicas ξ_a, ξ_by ξ_c, que definen las condiciones de terminación son

$\xi_a = 0$ mientras que barco permanezca dentro de las fronteras $-2 < \mathrm{x} < 10, -5 < \mathrm{y} < 5$;
$\xi_b = 0$ mientras que el barco no cruce los ejes $\mathrm{x} = 0$ ó $\mathrm{y} = 0$;
$\xi_c = 0$ hasta tres períodos de muestreo.



(a) (b)

Figura 2. Resultados de 10^6 repeticiones del Algoritmo 1 para el ejemplo del barco dirigido. a) Trayectoria del barco. b) Espacio de estado explorado.

En la Fig.2 se muestran los resultados obtenidos para un millón de pruebas. Como se puede observar, la trayectoria obtenida, indicada en la Fig.2a (que insume 2025 segundos para arribar a la meta), dista bastante de ser una trayectoria óptima. Simplemente, aplicando el Algoritmo 1 con ese elevado número de pruebas fue posible obtener una política mínimamente aceptable para guiar el barco desde el punto de partida hasta la meta. Los puntos marcados con círculos verdes son los estados en los que se producen los cambios de modo. En la Fig. 2b se

muestra todos los estados visitados durante el aprendizaje, lo cual define la región de interés del espacio de estado, compuesta por 138000 estados diferentes. A partir de esta gráfica se pueden observar dos cuestiones importantes. Primero, la elevada cantidad de estados que se necesitan para obtener una función de valor que permita obtener una política que pueda alcanzar el objetivo. Segundo, es notorio que existe una exploración del espacio de estados notoriamente grande.

4. Programación dinámica basada en simulación con muestreo de Lebesgue

La PD basada en simulación es una metodología de solución para problemas de decisión secuencial, que integra retroalimentación evaluativa y algoritmos de RL (Sutton y Barto 1998; Deisenroth, Rasmussen, y Peters 2009; Busoniu et al. 2010) para aprender a lograr una meta (tarea de control) mediante interacciones simuladas entre el agente que toma decisiones (controlador) y un sistema o entorno sobre el que influye. Al recurrir a la PD basada en simulación para la operación óptima de un sistema controlado, el objetivo es aprender una política de conmutación óptima, π^*, que define qué modo de control debe aplicarse para cualquier par estado/perturbación, teniendo en cuenta tanto las recompensas a corto plazo como las a largo plazo.

Algoritmo 2. $\mathcal{L}$DP.

1: **Input**: $\gamma, N, \mathcal{M}_{\mathrm{DP}}$
2: $\mathcal{X}_k^{\mathcal{L}}, k = 0, \dots, N \leftarrow$ Algoritmo 3
3: $V^*(\mathcal{X}_N^{\mathcal{L}}) = r(N)$
4: **For** k=N -1 to 0 **do**
5: **For** all $\mathbf{z}_i \in \mathcal{X}_k^{\mathcal{L}}$ **do**
6: **For** all $\boldsymbol{\Sigma}_j \in \mathcal{M}_{\mathrm{DP}}$ **do**
7: $Q(\mathbf{z}_i, \boldsymbol{\Sigma}_j) = r(k) + \gamma\, E[V_{k+1}(\mathbf{z}_{k+1}) | \mathbf{z}_i, \boldsymbol{\Sigma}_j]$
8: **End for**
9: $\pi^*(\mathbf{z}_i) \in \arg\max_{\boldsymbol{\theta}} Q(\mathbf{z}_i, \boldsymbol{\Sigma}_j)\,;\ \boldsymbol{\Sigma}_j \in \mathcal{M}_{\mathrm{DP}}$
10: $V_k^*(\mathbf{z}_i) = Q(\mathbf{z}_i, \pi^*(\mathbf{z}_i))$
11: **End for**
12: **End for**
13: **Return** $\pi^*(\mathcal{X}_k^{\mathcal{L}}), k = 0, \dots, N$

Tabla 2. Algoritmo de programación dinámica con muestreo de Lebesgue.

En la Tabla 2 se propone el Algoritmo 2 -$\mathcal{L}$DP- de PD que aplica un muestreo de Lebesgue al EE. Inicialmente, para una dinámica dada de perturbación $\dot{\mathbf{d}}(\mathrm{t})$ y asumiendo que se tiene un conjunto finito de modos $\mathcal{M}_{\mathrm{DP}}$, se genera un conjunto de estados alcanzables $\mathcal{X}^{\mathcal{L}} = \mathcal{X}_1^{\mathcal{L}} \cup \mathcal{X}_2^{\mathcal{L}} \dots \cup \mathcal{X}_N^{\mathcal{L}}$ empleando el Algoritmo 3 de la Tabla 3. Este conjunto de estados alcanzables aproxima un espacio alcanzable $\mathbb{X}^{\mathcal{L}}$. Posteriormente, el algoritmo $\mathcal{L}$DP calcula recursivamente el modo de control usando la política de conmutación óptima π*. Esto es, comenzando desde la etapa terminal N, $\mathcal{L}$DP explota el principio de optimalidad de Bellman para determinar la función de valor $\mathrm{V}^*(\mathcal{X}_N^{\mathcal{L}})$ y el correspondiente parámetro de modo óptimo $\pi^*(\mathcal{X}_N^{\mathcal{L}})$. Para cualquier estado $\mathbf{x}(\mathrm{t}) \in \mathcal{X}_N^{\mathcal{L}}$, la función de valor es inicializada usando $r(N)$.

En la línea 7 del Algoritmo 2 (Tabla 2), los valores Q^* son calculados recursivamente para cualquier $(\mathbf{x}(k), \mathbf{d}(k), \boldsymbol{\Sigma}(k))$. Cabe señalar que para una dinámica dada de la perturbación, el estado sucesor de un modo de control elegido de $\mathcal{M}_{\mathrm{DP}}$ estará siempre incluido en $\mathcal{X}^{\mathcal{L}}$. Cada parámetro óptimo $\pi^*(\mathbf{x}(k), \mathbf{d}(k))$, en la etapa de recursión k, es el argumento de maximización de los valores Q^* para todos los modos que pueden ser elegidos en un particular par $\mathbf{z}(k) = (\mathbf{x}(k), \mathbf{d}(k))$, y la función de valor V_k^* es el valor máximo correspondiente. Sin embargo, una pequeña advertencia, acerca de la optimalidad de la política resultante $\pi^*(\mathcal{X}_N^{\mathcal{L}})$ es que depende de cuán bien el conjunto $\mathcal{X}_N^{\mathcal{L}}$ aproxima al espacio alcanzable $\mathbb{X}^{\mathcal{L}}$. Como el Algoritmo 2 - $\mathcal{L}$DP -, de la Tabla 2, está basado en un número finito de modos de control en $\mathcal{M}_{\mathrm{DP}}$, la política de control resultante es una mera aproximación a la política óptima, la cual sólo podría obtenerse si se considera un dominio continuo en la definición de los modos de control.

Algoritmo 3. Lebesgue sampling
Input: $\mathcal{X}_0^{\mathcal{L}}, \mathcal{M}_{DP}, \dot{\mathbf{d}}(t), \mathbf{d}_0, t_0, L$.
$\mathcal{X}^{\mathcal{L}} = \emptyset$; $\mathcal{X}_k^{\mathcal{L}} = \emptyset, k = 1, \dots, N$
For all $\boldsymbol{\Sigma}_i \in \mathcal{M}_{DP}$,
  For all $\mathbf{x}_0 \in \mathcal{X}_0^{\mathcal{L}}$
    $\mathbf{x}(t(1,\boldsymbol{\Sigma}_i), \boldsymbol{\Sigma}_i) = \int_{t_0}^{t(1,\boldsymbol{\Sigma}_i)} f(\mathbf{x}(t), (\mathcal{K}(\mathbf{x}), \boldsymbol{\xi}(\mathbf{x}(t), \mathbf{d}(t))))dt\,,\ \mathbf{x}(t_0) = \mathbf{x}_0$
    $\mathbf{d}(t(1,\boldsymbol{\Sigma}_i)) = \int_{t_0}^{t(1,\boldsymbol{\Sigma}_i)} \dot{\mathbf{d}}(t)\, dt,\ \mathbf{d}(t_0) = \mathbf{d}_0$
    $\mathbf{z}(1) = (\mathbf{x}(t(1,\boldsymbol{\Sigma}_i), \boldsymbol{\Sigma}_i), \mathbf{d}(t(1,\boldsymbol{\Sigma}_i)))$
    $\mathcal{X}_1^{\mathcal{L}} = \mathcal{X}_1^{\mathcal{L}} \cup \mathbf{z}(1)$
  End For
End For
$p = 1$
Repeat
  $p = p + 1$
  $k = uniform(1, N-1)$
  $\tilde{\boldsymbol{z}}(k) = rand(\mathbf{z} \in \mathcal{X}_k^{\mathcal{L}})$
  $\boldsymbol{\Sigma}_j := rand(\mathcal{M}_{DP})$

$\mathbf{x}(t(k+1, \boldsymbol{\Sigma}_j), \boldsymbol{\Sigma}_j) = \int_{t(k,\boldsymbol{\Sigma}_j)}^{t(k+1,\boldsymbol{\Sigma}_j)} f(\mathbf{x}(t), (\mathcal{K}(\mathbf{x}), \boldsymbol{\xi}(\mathbf{x}(t), \mathbf{d}(t))))dt$
  $\mathbf{d}(t(k+1, \boldsymbol{\Sigma}_j)) = \int_{t(k,\boldsymbol{\Sigma}_j)}^{t(k+1,\boldsymbol{\Sigma}_j)} \dot{\mathbf{d}}(t)\, dt$
  $\mathbf{z}(k+1) = (\mathbf{x}(t(k+1, \boldsymbol{\Sigma}_j), \boldsymbol{\Sigma}_j), \mathbf{d}(t(k+1, \boldsymbol{\Sigma}_j)))$
  If $\mathbf{z}(k+1) \notin \mathcal{X}_{k+1}^{\mathcal{L}}$
    $\mathcal{X}_{k+1}^{\mathcal{L}} = \mathcal{X}_{k+1}^{\mathcal{L}} \cup \mathbf{z}(k+1)$
  End If
Until $mod(p, L) = 0$
Output: $\mathcal{X}^{\mathcal{L}} = \mathcal{X}_1^{\mathcal{L}} \cup \dots \cup \mathcal{X}_N^{\mathcal{L}}$

Tabla 3. Algoritmo de generación del conjunto de estados alcanzables $\mathcal{X}^{\mathcal{L}}$.

4.1. Ejemplo de guiado del barco (continuación)

Se aplicó el algoritmo $\mathcal{L}$DP propuesto que se introduce en la Tabla 2. Se definieron los modos de igual manera que en la aplicación anterior, en el que todos ellos responden a una misma ley de control: $\mathcal{K}(\mathbf{x}) = \phi$ y el parámetro ϕ puede tomar valores en un conjunto discretizado en $[0°, 360°]$. Si un valor de ϕ es seleccionado por el controlador, el mismo deberá permanecer constante hasta que se active una condición de terminación.

Las condiciones de terminación son iguales para todos los modos y las mismas se definen a través de un vector binario $\boldsymbol{\xi}_i$, tal que $\boldsymbol{\xi}_i = 0 \Leftrightarrow \xi_a = \xi_b = \xi_c = 0$; de lo contrario $\boldsymbol{\xi}_i = 1$. El Algoritmo $\mathcal{L}$DP incorpora estimaciones de los valores de función de estado ($V_k(\mathbf{x})$), por tanto, a diferencia del caso anterior (ejemplo en Sección 3.3), en esta aplicación se utilizan las estimaciones de las funciones de valor en las condiciones de terminación. Esto posibilita una mejor exploración del espacio de estado, otorgando la posibilidad de elegir entrar en aquellos con mayor valor de función. Por esta razón se produce un muestreo "más natural" del espacio de estado capitalizando la experiencia recogida durante la ejecución del algoritmo y sintetizada en la función de valor V. Un beneficio adicional radica, por ejemplo, en que ya no será necesario fijar un máximo para la cantidad de

períodos de muestreo que un modo puede permanecer activo. Entonces, las funciones para las condiciones de terminación son
$\xi_a = 0$ mientras que el barco permanezca dentro de la región $-2 < \mathrm{x} < 10, -5 < \mathrm{y} < 5$; de lo contrario $\xi_a = 1$;
$\xi_b = 0$ mientras que el barco no cruce los ejes $\mathrm{x} = 0$ ó $\mathrm{y} = 0$; de lo contrario $\xi_b = 1$.
$\xi_c = 0$ siempre que $\nabla V(\mathbf{x}) > 0$; de lo contrario $\xi_c = 1$.

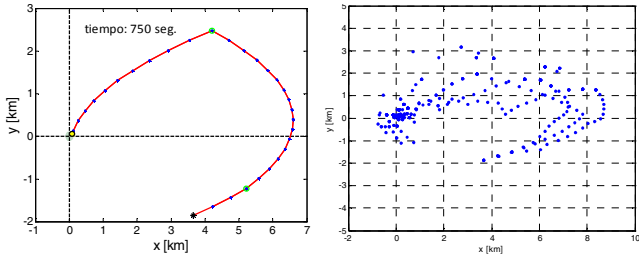


Figura 3. Resultado de aplicar $\mathcal{L}$DP al ejemplo del barco dirigido. a)Trayectoria del barco. b) Espacio de estado explorado.

En la Fig. 3 se muestra el resultado de aplicar el algoritmo $\mathcal{L}$DP, donde se ve la trayectoria obtenida (Fig. 3a) aplicando la política óptima del Algoritmo 2, la cual logra conducir el barco hacia la meta en solo 750 segundos. Esto revela que la propuesta aquí realizada consigue una política multimodal más eficiente que la dada por el Algoritmo 1 (Sección 3). Por otra parte, vale la pena destacar que sólo 3500 estados (Fig. 3b) fueron incluidos por el algoritmo para conseguir una representación de la zona de interés alcanzable del espacio de estado, sobre la cual se determina la política óptima de conmutación de modos.

5. Control multimodal óptimo bajo incertidumbre

Mediante el enfoque presentado en la sección anterior, es posible encontrar una solución a la tarea de control multimodal. No obstante, existen dificultades e inconvenientes que se traducen en limitaciones para enfrentar tareas más complejas, como lo es la existencia de un entorno incierto, que en el caso del ejemplo del barco dirigido bien podría considerarse la corriente de agua con comportamiento variable, asemejandose más a una situación real.

El principal inconveniente con $\mathcal{L}$DP es que cuando las transiciones de estados están sujetas a incertidumbre aparecen estados no visitados y por tanto desconocidos. Para evitar este problema, $\mathcal{L}$DP debería ser combinado con una técnica de regresión para generalizar la política de conmutación π a estados que no pertenecen necesariamente a la vecindad de estados conocidos $\mathbf{x}(t) \in \mathcal{X}^{\mathcal{L}}$. También, la búsqueda de una política de conmutación cuasi óptima bajo incertidumbre debe muestrear selectivamente transiciones de estado usando sólo los modos de control más promisorios para conducir el sistema desde un estado inicial hacia uno objetivo, requiriendo un bajo número de corridas de simulación. Esto impone la necesidad de introducir aprendizaje activo en la estrategia de muestreo de estados. De esta forma, es posible focalizarse en el mejoramiento de π mediante la generación de las secuencias $(\mathbf{x}^{\pi}, \mathbf{d}, \mathbf{t}^{\pi}, \boldsymbol{\theta}^{\pi})$ más informativas dando lugar a un sesgo incremental en el muestreo de los datos. Es de destacar que en un entorno incierto, a medida que se modifica iterativamente la política, se visitan nuevos estados que generan la necesidad de seleccionar aquellos que aportan la mejor información para la aproximar las funciones de valor.

5.1. Programación dinámica con muestreo de Lebesgue, procesos Gaussianos y aprendizaje activo

Procesos Gaussianos

Un proceso Gaussiano es un ejemplo de la utilización de un modelo probabilístico, flexible y no-paramétrico que permite directamente realizar predicciones con una caracterización de la incertidumbre. Su uso y propiedades están profusamente tratadas en los trabajos de Rasmussen y Williams (2006), Kuss (2006) y Girard (2004).

En un problema de aprendizaje la experiencia asociada con la secuencia de interacciones entre el agente y su entorno se recopila en conjuntos de datos de entrenamiento ("*training set*") de la forma $\mathfrak{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$, tal que el conjunto de vectores de entrada ("*training inputs*") es $\boldsymbol{X} : \{\mathbf{x}_i \in \mathbb{R}^m \,/\, i = 1, 2 \ldots n\}$ y su correspondiente conjunto de observaciones ("*training targets*") es $\boldsymbol{Y} : \{y_i \in \mathbb{R}/i = 1,2 \ldots n\}$; donde n es el número de interacciones experimentadas por el agente. Ahora, si se considera que la relación entre los datos de entrada-salida puede expresarse mediante una relación funcional h, asumiendo que los mismos pueden ser ejemplos representativos de la relación $y_i = h(\mathbf{x}_i) + \varepsilon_i$, donde $h: \mathbb{R}^m \to \mathbb{R}$ y el ruido o error de medición $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ se asume independiente de $h(\mathbf{x}_i)$; para hacer inferencias acerca del valor de la salida y^* para un punto elegido del dominio, $\mathbf{x}^*$, se necesita un modelo inferencial para la función subyacente h.

A *priori*, la función subyacente h es considerada como una función aleatoria. Así, un proceso Gaussiano se define como una colección de valores aleatorios $\boldsymbol{H} = h(\boldsymbol{X}) = \{h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n)\}$, con distribución Gaussiana conjunta. De este modo, un proceso Gaussiano es una generalización de la distribución de probabilidad Gaussiana, es decir un GP es una distribución sobre funciones. Análogamente a una distribución Gaussiana, la cual queda completamente especificada por el vector de media y la matriz de covarianza, un GP queda completamente especificado por una función de media $m(\mathbf{x})^*$ y función de covarianza $Cov(\mathbf{x}, \mathbf{x}')$, tal que:

$$\begin{aligned} m(\mathbf{x}) &= E[h(\mathbf{x})] \\ Cov(\mathbf{x}, \mathbf{x}') &= E\left[\left(h(\mathbf{x}) - m(\mathbf{x})\right)\left(h(\mathbf{x}') - m(\mathbf{x}')\right)\right] \end{aligned} \tag{7}$$

donde $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^m$; entonces, usando las nociones de función de media y covarianza, un GP puede ser matemáticamente expresado como:

$$h(x) = \mathcal{GP}(m(\mathbf{x}), Cov(\mathbf{x}, \mathbf{x}')) \tag{8}$$

entonces, para la colección de valores aleatorios $\boldsymbol{H} = h(\boldsymbol{X})$, la distribución Gaussiana conjunta se puede escribir como: $p(h(\boldsymbol{X})) = \mathcal{N}(m(\boldsymbol{X}), Cov(\boldsymbol{X}, \boldsymbol{X}))$, dónde $m(\boldsymbol{X})$ es el vector de valores medios y $\boldsymbol{K} = Cov(\boldsymbol{X}, \boldsymbol{X})$ es la matriz de covarianza de todos los valores $\boldsymbol{H} = h(\boldsymbol{X})$ bajo consideración. Téngase en cuenta que la función de covarianza $Cov(\cdot, \cdot)$, puede ser cualquier función con la propiedad para generar una matriz de covarianza $\boldsymbol{K}$ positiva definida. En este trabajo se recurre a la comúnmente empleada función Gaussiana:

$$Cov(\mathbf{x}_p, \mathbf{x}_q) = \alpha^2 \exp\left(-\tfrac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^{\mathrm{T}} \Lambda^{-1} (\mathbf{x}_p - \mathbf{x}_q)\right) \tag{9}$$

*Obsérvese que una función de media $m(\cdot)$ se representa siempre con argumento, en tanto cuando se usa m sin argumento, corresponde a la dimensión de los vectores $\mathbf{x}$.

donde el parámetro α^2 describe la variabilidad de la función subyacente h y $\Lambda = diag([l_1^2, \dots, l_{n_x}^2])$ siendo $l_r, r = 1, \dots, n_x$ los parámetros de escala. Todos estos parámetros representan lo que se denomina como los "hyperparámetros" del modelo GP, que pueden ser agrupados en un vector $\boldsymbol{\psi}$. Otras ejemplos de funciones de covarianza pueden encontrarse en libro de Rasmussen y Williams (2006) para ser empleadas en diferentes aplicaciones. Normalmente, para un problema dado los hyperparámetros se determinan mediante la maximización de la evidencia respecto al conjunto de datos disponible (datos de entrenamiento). Cuando el conjunto de datos es demasiado grande esta tarea suele ser computacionalmente demandante dado que se debe calcular la inversa de la matriz de covarianza de los datos.

Una vez que los hyperparámetros han sido obtenidos, la distribución predictiva o *a posteriori* Gaussiana de un valor de salida $y^* = h(\mathbf{x}^*)$ para un punto arbitrario $\mathbf{x}^*$ queda especificada por la media y covarianza:

$$E\left[h\left(\mathbf{x}^*\right) | \boldsymbol{X}, \boldsymbol{Y}, \mathbf{x}^*\right] = Cov\left(\mathbf{x}^*, \boldsymbol{X}\right)\left[Cov(\boldsymbol{X}, \boldsymbol{X}) + \sigma_w^2 I\right]^{-1} \boldsymbol{Y} \tag{10}$$

$$Cov(h(\mathbf{x}^*)) = Cov(\mathbf{x}, \mathbf{x}^*) - Cov(\mathbf{x}^*, \boldsymbol{X})[Cov(\boldsymbol{X}, \boldsymbol{X}) + \sigma_w^2 I]^{-1} Cov(\boldsymbol{X}, \mathbf{x}^*) \tag{11}$$

Para efectuar predicciones k-pasos hacia el futuro, es necesario contemplar la incertidumbre de las futuras predicciones. Basándose en el trabajo de Girard y Colab. (2004), es posible usar una aproximación gaussiana a la incertidumbre de las entradas. La distribución predictiva o a *posteriori*, para la salida correspondiente al punto de entrada aleatorio y arbitrario $\mathbf{x}^*$ es exactamente:

$$p\left(h\left(\mathbf{x}^*\right) | \mu, \Sigma\right) = \int p\left(h\left(\mathbf{x}^*\right) | \mathbf{x}^*\right) p(\mathbf{x}^* | \mu, \Sigma) d\mathbf{x}^* \tag{12}$$

la cual no es una distribución Gaussiana. La misma se puede aproximar por una distribución con media:

$$\mu^* = E_h\left[E_{\mathbf{x}^*}\left[h\left(\mathbf{x}^*\right)\right]\right] = E_{\mathbf{x}^*}\left[E_h\left[h(\mathbf{x}^*)\right]\right] = E_{\mathbf{x}^*}\left[m(\mathbf{x}^*)\right] = \int m(\mathbf{x}^*) p(\mathbf{x}^*) d\mathbf{x}^* = \boldsymbol{\beta}^T \boldsymbol{l} \tag{13}$$

siendo $\boldsymbol{\beta} = (\boldsymbol{K} + \sigma_w^2 \boldsymbol{I})^{-1} \boldsymbol{Y}$ y $l_i = \int Cov(\mathbf{x}_i, \mathbf{x}^*)\, p(\mathbf{x}^*)\, d\mathbf{x}^* = \alpha^2 \, |\Sigma\Lambda^{-1} + \boldsymbol{I}|^{-1/2} \, \exp\left(-\frac{1}{2}\,(\mathbf{x}_i - \mu)^{\mathrm{T}}\,(\Sigma + \Lambda^{-1})\,(\mathbf{x}_i - \mu)\right)$ y la varianza $\mathcal{V}^2$ de la distribución predictiva está dada por:

$$\mathcal{V}^2 = E_{\mathbf{x}^*}\left[m\left(\mathbf{x}^*\right)^2\right] + E_{\mathbf{x}^*}\left[Cov\left(\mathbf{x}^*, \mathbf{x}^*\right)^2\right] - E_{\mathbf{x}^*}\left[m\left(\mathbf{x}^*\right)\right]^2 = \boldsymbol{\beta}^T \boldsymbol{L} \boldsymbol{\beta} + \alpha^2 - tr\left(\left(\mathbf{K} + \sigma_w^2 \boldsymbol{I}\right)^{-1} \boldsymbol{L}\right) - \mu^{*2} \tag{14}$$

dónde: $L_{ij} = Cov(\mathbf{x}_i, \mu) Cov(\mathbf{x}_j, \mu) / |2\Sigma\Lambda^{-1} + I|^{-1/2} \exp\left[(\tilde{\mathbf{z}}_{ij} - \mu)^T \left(\Sigma + \frac{1}{2}\Lambda\right)^{-1} \Sigma\Lambda^{-1} (\tilde{\mathbf{z}}_{ij} - \mu)\right]$; siendo $\tilde{\mathbf{z}}_{ij} = \frac{1}{2}(\mathbf{x}_i + \mathbf{x}_j)$ y $tr(\boldsymbol{M})$ las trazas de una dada matriz $\boldsymbol{M}$. Obsérvese que la media μ^* y varianza $\mathcal{V}^2$ predictivas dependen explícitamente de la media, μ, y la matriz de covarianza, Σ, del punto de entrada incierto $\mathbf{x}^*$.

Formulación algorítmica de programación dinámica con muestreo de Lebesgue y modelos de Procesos Gaussianos

En un entorno sujeto a incertidumbre ocurren transiciones de estados que pueden conducir a estados no visitados previamente y por tanto desconocidos. En consecuencia, ya no es posible realizar una discretización finita del espacio de estado dado que se debe enfrentar una problemática con un continuo de estados alcanzables, por lo que no es viable el empleo de una representación tabular para las funciones de valor. Por ello, para aproximar las funciones de valor se utilizan los modelos GPs indicados anteriormente, lo que posibilita realizar una formulación para trabajar en un dominio continuo de estados en vistas a encontrar una política de control óptima a través de la resolución del problema de programación dinámica. Por otra parte, la política de control, la cual es una función que mapea estados-acciones, puede ser también modelada mediante modelos de GPs. Usar GPs para representar la política aporta una característica distintiva dado que cada modo inferido, por el modelo GP de la política, tendrá asociado un grado de certeza caracterizado por la desviación estándar de la predicción realizada.

En el algoritmo de la Tabla 4, que se denomina como ℒGPDP, se presenta una versión que consiste en una generalización del algoritmo ℒDP (de la Tabla 2). ℒGPDP permite hallar una solución para problemas donde se tienen espacios continuos de estados y además permite trabajar con un continuo de modos de control parametrizados. Para ello, los modos de control se definen como una ley de control de retroalimentación parametrizada y condiciones de terminación con parámetros libres. Por tanto, una política óptima debe determinar en cada momento de conmutación de modo un vector óptimo de parámetros libres.

Algoritmo 4. ℒGPDP.

Input: $\gamma, N, \mathfrak{X}_k^S, k = 1, \dots, N$ and Θ^S
$V^*(\mathfrak{X}_N^S) = r(N)$
$V_N^* \sim \mathcal{GP}_v$ ⇒ modelo GP para $\mathcal{GP}_v$
For $K = N - 1$ to 0 **do**
  For all $\mathbf{z}_i \in \mathfrak{X}_k^S$ **do**
    For all $\boldsymbol{\theta}_j \in \Theta^S$ **do**
      $Q(\mathbf{z}_i, \boldsymbol{\theta}_j) = r(k) + \gamma\, E[\mathcal{GP}_v(\mathbf{z}_{k+1}) | \mathbf{z}_i, \boldsymbol{\theta}_j]$
    End for
    $Q_k^*(\mathbf{z}_i, \cdot) \sim \mathcal{GP}_q$ ⇒ modelo GP para Q_k^*
    $\pi^*(\mathbf{z}_i) \in \arg\max_\theta\, Q(\mathbf{z}_i, \boldsymbol{\theta}_j)\,;\ \boldsymbol{\theta}_j \in \Theta^S$
    $V_k^*(\mathbf{z}_i) = Q(\mathbf{z}_i, \pi^*(\mathbf{z}_i))$
  End for
  $V_k^*(\cdot) \sim \mathcal{GP}_v$ ⇒ modelo GP para $V_k^*(\cdot)$
End for
Aproximar $\pi^* \sim \mathcal{GP}_{\pi^*}$ con $\mathfrak{X}_k^S$ y $\pi^*(\mathfrak{X}_k^S), k = 1, \dots, N$
Return $\mathcal{GP}_v$, $\pi^*(\mathfrak{X}_k^S)$, π^*

Tabla 4. Algoritmo de programación dinámica con muestreo de Lebesgue y aproximaciones con modelos de GPs.

El algoritmo ℒGPDP emplea modelos probabilísticos de GPs para describir las funciones de valor $V_k^*(\cdot)$ y $Q_k^*(\cdot,\cdot)$ directamente en el espacio de funciones mediante la representación de las mismas a través de modelos probabilísticos de GPs que utilizan datos reales (o simulados) para determinar las estructuras subyacentes de las funciones de valor, las cuales son *a priori* desconocidas. Además, estos modelos proporcionan información acerca de los intervalos de confianza para las estimaciones de funciones de valor y el ajuste cuasi óptimo de los parámetros del modo de control que debe ser implementado. Similarmente al algoritmo ℒDP de la Tabla 2, ahora se seleccionan los conjuntos finitos $\mathfrak{X}_k^S, k = 1, \dots, N$ y Θ^S para estados y parámetros de modos $\boldsymbol{\theta}_j \in \Theta^S$, respectivamente. Los conjuntos $\mathfrak{X}_k^S$ y Θ^S son considerados como *conjuntos de soporte* (datos de entrenamiento) para aproximar las funciones de valor $V_k^*(\cdot)$ y $Q_k^*(\cdot,\cdot)$ usando modelos de GPs, tal que: $V_k^*(\cdot) \sim \mathcal{GP}_v(m_v, Cov_v)$ y $Q_k^*(\cdot,\cdot) \sim \mathcal{GP}_q(m_q, Cov_q)$, respectivamente.

Los valores "targets" en el entrenamiento (observaciones) para $\mathcal{GP}_v$ y $\mathcal{GP}_q$ son determinados recursivamente por el mismo ℒGPDP. La ventaja de modelar la función de valor de estados

$V_k^*(\cdot)$ mediante $\mathcal{GP}_v$ es que el modelo GP ofrece una distribución predictiva de $V_k^*(\mathbf{z}_k)$ para *cualquier* entrada $\mathbf{z}_k$, a través de las ecuaciones (7) y (8). Esta propiedad es explotada en el cálculo del valor de Q_k^* (línea 7), debido a la propiedad de generalización de $\mathcal{GP}_v$ que no está restringida a un conjunto finito de estados sucesores cuando se debe determinar $E_v[\mathcal{GP}_v(\mathbf{z}_{k+1})]$, luego de la ejecución de un modo. Es para destacar que el uso de un modelo probabilístico para la función de valor permite hacer frente de una manera natural a la incertidumbre, tanto en las transiciones de estados, producidas por la ejecución de un modo de control, como a la variabilidad en las recompensas. Como el operador de la esperanza matemática depende de la función subyacente de $V_{k+1}^*(\mathbf{z}_{k+1})$, la cual es probabilísticamente modelada mediante $\mathcal{GP}_v$, $E_v[\mathcal{GP}_v(\mathbf{z}_{k+1})]$ se puede tomar simplemente como la $m_v(\mathcal{GP}_v(\mathbf{z}_{k+1}))$, es decir la media de predicción de $V_{k+1}^*(\mathbf{z}_{k+1})$. Nótese que para cualquier $\mathbf{z}_k \in \mathfrak{X}_k^S$, el correspondiente $\mathcal{GP}_q$ modela una función sobre un continuo de parámetros de modos y no solamente lo hace sobre el conjunto soporte finito Θ^S. Por tanto, la parametrización óptima del modo de control óptimo para $\mathbf{z}_k$, en la línea 10 es el máximo de la función de media de $\mathcal{GP}_q$. Entonces, la parametrización para el modo de control óptimo para un $\mathbf{z}_k$ se obtiene resolviendo el problema de optimización como el indicado en la ecuación (15), usando métodos convencionales de optimización.

$$\begin{aligned} &\arg\max_{\boldsymbol{\theta}} Q_k^*(\mathbf{z}_k, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} m_q(\boldsymbol{\theta}) \\ &\text{s.a} \qquad \boldsymbol{\theta}^{\min}(\mathbf{z}_k) \le \theta \le \boldsymbol{\theta}^{\max}(\mathbf{z}_k) \end{aligned} \tag{15}$$

Una característica distintiva de la optimización dinámica basada en simulación realizada con $\mathcal{L}$GPDP, es que para todo $\mathbf{z}_k \in \mathfrak{X}_k^S$ son usados modelos independientes de GP para $Q_k^*(\mathbf{z}_k,\cdot)$ en lugar de modelar $Q_k^*(\cdot,\cdot)$ en un espacio común de estados-acciones como proponen Mehta y Egerstedt (2006). La principal razón para esta elección es que, por medio de un modelo de simulación para las transiciones de estados, el algoritmo $\mathcal{L}$GPDP solamente necesita la función de valor $V_k^*(\mathbf{z}_k)$, entonces el máximo esperado de recompensa acumulada en un punto de muestreo de Lebesgue es estimado usando $\mathcal{GP}_v$. Asimismo, un buen modelo de la función Q_k^* en el espacio común de estado-acción requiere sustancialmente más puntos de entrenamiento, lo que hace que los modelos estándar de GP se vuelvan computacionalmente muy costosos. En contraste con la PD clásica, $\mathcal{L}$GPDP es independiente de la frecuencia espacio-temporal de muestreo.

Cuando se comparan con $\mathcal{L}$DP, los conjuntos $\mathfrak{X}_k^S, k = 1, \ldots N$ en $\mathcal{L}$GPDP contienen puntos de soporte de los modelos GP de la función de valor en lugar de una representación discreta $\mathcal{X}^{\mathcal{L}}$ del espacio de estados alcanzable $\mathbb{X}^{\mathcal{L}}$. Así, por medio del muestreo de Lebesgue, $\mathcal{L}$GPDP puede utilizar los datos mucho más eficientemente que la versión con discretización de Riemman de GPDP propuesta por Deisenroth y Colab. (2009). También, recurriendo a modelos inductivos del tipo GPs, el algoritmo propuesto $\mathcal{L}$GPDP puede hacer frente a la incertidumbre en las transiciones de estados debido a la dinámica estocástica de las perturbaciones y a los ruidos presentes en las mediciones.

Para generalizar una política de conmutación de modos óptima en un espacio de estados alcanzables $\mathbb{X}^{\mathcal{L}}$, se debe modelar la política basándose en el conjunto de soporte $\mathfrak{X}_k^S, k = 1, \ldots N$. La política de conmutación π^* es considerada como un mapeo determinístico entre estados/perturbaciones a parametrizaciones de modo de control. Para generalizar la política de conmutación basándose en un conjunto finito de parámetros de los modos de control Θ^S, es necesario resolver un problema de regresión que permite disponer de un modelo inductivo de la política de conmutación óptima π^* válida sobre todo el espacio $\mathbb{X}^{\mathcal{L}}$. El conjunto de soporte $\mathfrak{X}_k^S, k = 1, \ldots N$, contiene los puntos de entrenamiento para el modelo GP de la política, que a la vez son los puntos de entrenamiento del modelo GP de la función de valor; en tanto que los *targets* de entrenamiento son los valores de los parámetros $\pi^*(\mathfrak{X}_k^S)$. Si se carece de especificaciones a priori del problema, este enfoque general es de alta eficacia. Aunque cualquier técnica de aproximación de función puede ser usada para modelar la política, en este trabajo la misma es aproximada mediante un GP (línea 15 del Algoritmo 4). En general, encontrar una política global óptima es muy dificultoso; además se requieren muchos datos, particularmente en espacios de estados de gran dimensión. En las aplicaciones prácticas, no se requiere una política globalmente óptima, sino más bien una política que pueda resolver una particular tarea de control de forma eficiente.

Hasta el momento, se han incluido los puntos de soporte $\mathfrak{X}_k^S, k = 1, \ldots N$ para el modelo de la función de valor $\mathcal{GP}_v$ basado en muestreo del espacio de estados/perturbaciones usando secuencias aleatorias de modos de control. Esta estrategia así planteada es muy subóptima, pudiendo ser altamente ineficiente respecto al volumen de datos necesarios para lograr una cobertura significativa del espacio de estado a medida que aumenta el número de etapas de decisión. Sin embargo, la falta de conocimiento previo acerca de una política de conmutación cuasi óptima y con el fin de lograr un adecuado equilibrio entre la exploración y explotación, el muestreo del espacio de estados debe limitarse únicamente a la región relevante. Para ello, en lo que resta de esta sección se describirá como combinar de manera *on-line* un modelado de la dinámica de transiciones de estados usando GPs y aprendizaje Bayesiano activo para explorar una región relevante del espacio de estados, empleando una función de utilidad para destacar los puntos relevantes de dicho espacio.

Formulación algorítmica de PD con muestreo de Lebesgue y un aprendizaje activo

Para encontrar una política de conmutación óptima en el Algoritmo 5 de la Tabla 5 se incorpora el mecanismo de aprendizaje Bayesiano activo para que sólo sea explorada la porción relevante del espacio de estado. A priori no es claro que partes del espacio de estado son relevantes. La idea de "relevancia" es caracterizada mediante la función de utilidad

$$\mathfrak{U}(\tilde{\mathbf{z}}) = \rho E_v\left[V^*(\tilde{\mathbf{z}}) \,|\, \mathfrak{X}_{k-1}\right] + \frac{\beta}{2}\log\left(\mathrm{var}_v\left[V^*(\tilde{\mathbf{z}}) \,|\, \mathfrak{X}_{k-1}\right]\right) \tilde{\mathbf{z}} \in \tilde{\mathfrak{X}} \tag{16}$$

en la que la distribución a posteriori del modelo $\mathcal{GP}_v$ de la función de valor juega un rol preponderante. Este mecanismo explota en gran medida la información ya calculada dentro del propio algoritmo $\mathcal{L}$GPDP. De esta manera, la combinación del aprendizaje Bayesiano activo y $\mathcal{L}$GPDP da lugar al Algoritmo 5 que se referirá como BA$\mathcal{L}$GPDP en la Tabla 5. En lugar de un modelo global y suficientemente preciso de la función de valor, BA$\mathcal{L}$GPDP pretende encontrar un modelo localmente apropiado para aproximar la función de valor en la vecindad de las trayectorias más promisorias, desde un estado inicial hacia otro final, usando una secuencia óptima de modos de control. De esta manera, durante el proceso de aprendizaje el algoritmo BA$\mathcal{L}$GPDP va sesgando la recopilación de los datos en busca de una solución, conduciendo el sistema desde un estado inicial hacia un estado final dentro una región relevante.

En la fase de aprendizaje Bayesiano activo, considérese un conjunto dado $\overline{\mathfrak{X}}_k^S$ de estados candidatos los cuales podrían ser incorporados al conjunto de soporte $\mathfrak{X}_k^S$ usado para el entrenamiento de los modelos GPs. Por razones de eficiencia, solamente los mejores candidatos deberían ser incorporados. En el control probabilístico óptimo un "buen" punto de entrada $\tilde{\mathbf{z}}^* \in \overline{\mathfrak{X}}_k^S$ debería proporcionar tanto la obtención de información acerca de la función de valor subyacente así como reducir la incertidumbre de las acciones de control óptimas para cada punto del área relevante del espacio de estado. Por consiguiente, se puede emplear una función de utilidad $\mathcal{U}(\cdot)$ como la de la ecuación (16) para evaluar la calidad de los estados usados con vistas a encontrar aquellos $\tilde{\mathbf{z}}^*$ más promisorios que la maximizan.

En la búsqueda de una mejora incremental de la política de conmutación óptima, el algoritmo BA$\mathcal{L}$GPDP explota el conocimiento actual representado y proporcionado por el modelo $\mathcal{GP}_v$ de la función de valor. Obtener información significa que BA$\mathcal{L}$GPDP debería explorar selectivamente subespacios escasamente muestreados con pocos puntos (estados) de entrenamiento. De esta forma, incorporando entradas $\tilde{\mathbf{z}}^*$ que son informativas para la operación óptima del sistema, las trayectorias de estados más promisorios desde un punto inicial a uno final pueden ser encontradas.

Algoritmo 5. BA$\mathcal{L}$GPDP.

1: **Input**: $\mathfrak{X}_0^S, \Theta^S, \delta, c, \gamma, \mathcal{T}, \vartheta$
2: Simular $\mathcal{T}$ trayectorias de longitud N aplicando π_0 a partir de $\mathfrak{X}_0^S \rightarrow$ $\rightarrow$ observe $\mathfrak{X} = \{\mathfrak{X}_k^S,\ k = 1, \dots, N\}$
3: Determinar $\mathcal{GP}_f$ con los datos simulados
4: $\pi_0^* = \pi_0$
5: Determinar $\mathcal{GP}_{\pi_0^*}$ con $\mathfrak{X}$ y con los parámetros $\boldsymbol{\theta} \in \Theta^S$ seleccionados por π_0^*
6: $p = 1$
7: $\mathrm{e} = \infty$
8: **Until** $\mathrm{e} < \delta$ **do**
9: Calcular $V_N^*(\mathbf{z}_N^1) = r(N)\ \forall\ \mathbf{z}_N^1 \in \mathfrak{X}_N^S \subset \mathfrak{X}$
10: Determinar $\mathcal{GP}_v$ con $\mathfrak{X}_N^S$ y V_N^*
11: **For** $k = N - 1$ to 0 **do**
12: Estimar $\overline{\mathfrak{X}}_{k+1}^S$ con $\mathcal{GP}_f$ from $\mathfrak{X}_k^S \subset \mathfrak{X}$
13: Determinar los ϑ estados más promisorios de $\overline{\mathfrak{X}}_{k+1}^S$ $\Rightarrow$ *aprendizaje Bayesiano*
14: Aumentar $\mathfrak{X}_k^S$ con los ϑ estados seleccionados de $\overline{\mathfrak{X}}_{k+1}^S$
15: Actualizar $\mathcal{GP}_f$ con el conjunto de datos aumentado $\mathfrak{X}_k^S$
16: **For** all $\mathbf{z}_i \in \mathfrak{X}_k^S$ **do**
17: **For** all $\boldsymbol{\theta}_j \in \Theta^S$ **do**
18: $Q(\mathbf{z}_i, \boldsymbol{\theta}_j) = r(k) + \gamma\ E[V_{k+1}(\mathbf{z}_{k+1}) |\ \mathbf{z}_i, \boldsymbol{\theta}_j, \mathcal{GP}_f]$
19: **End for**
20: $Q_k^*(\mathbf{z}_i, \cdot) \sim \mathcal{GP}_q$
21: $\pi^*(\mathbf{z}_i) \in \arg\max_\vartheta Q(\mathbf{z}_i, \boldsymbol{\theta}_j);\ \boldsymbol{\theta}_j \in \Theta^S$
22: $V_k^*(\mathbf{z}_i) = Q(\mathbf{z}_i, \pi^*(\mathbf{z}_i))$
23: **End for**
24: $V_k^*(\cdot) \sim \mathcal{GP}_v$
25: **End for**
26: $\mathfrak{X} = \mathfrak{X}_0^S \cup \mathfrak{X}_1^S \cup \dots \cup \mathfrak{X}_N^S$
27: Aproximar $\pi_p^* \sim GP_{\pi_p^*}$ con $\mathfrak{X}$ y $\pi^*(\mathbf{z}_i)\ \forall\ \mathbf{z}_i \in \mathfrak{X}$
28: $\pi_p^* := \mathcal{GP}_{\pi_p^*}$
29: **If** $p > 1$
30: $\mathrm{e} = \frac{\Re_{previous} - \Re_{now}}{\Re_{previous}} \cdot 100\%$
31: **End If**
32: $p = p + 1$
33: **End Loop**
34: **Return** $\pi^* := \pi_{p-1}^*$

Tabla 5. Algoritmo de programación dinámica con muestreo de Lebesgue, aproximaciones con modelos de GPs y aprendizaje Bayesiano activo.

En la Tabla 5 se indican los principales pasos del Algoritmo 5 BA$\mathcal{L}$GPDP. Primero un número pequeño (dígase $\mathcal{T}$) de trayectorias de estados/perturbaciones son simulados desde diferentes puntos iniciales $\mathbf{z}^1(0) \in \mathfrak{X}_0^S$ aplicando una secuencia de hasta N modos de control cuyos valores de parámetros se escogen aleatoriamente de un conjunto finito Θ^S basado en una política de conmutación inicial π_0. Las tuplas resultantes $\left(\mathbf{x}^1(k), \mathbf{d}^1(k), \boldsymbol{\theta}^1(k)\right)$, observadas a lo largo de las trayectorias simuladas, definen los datos de entrenamiento para el modelo GP de la dinámica en la primera iteración. Las transiciones de estado observadas naturalmente poseen una componente de ruido debido a la incertidumbre en la activación de las condiciones de terminación.

Para obtener la política de conmutación, en la primer iteración del algoritmo BA$\mathcal{L}$GPDP, una forma recursiva de PD se usa para aproximar $\mathcal{GP}_v$ comenzando con los estados finales $\mathbf{x}^1(N) \in \mathfrak{X}_N^S$ para los cuales la función de valor puede ser fácilmente calculada (línea 9) y luego usada para entrenar $V_N^*(\mathbf{z}_N^1)$ (línea 10). El bucle interno determina los conjuntos de soporte $\mathfrak{X}_k^S, k = N-1, \dots, 1$ usando aprendizaje Bayesiano activo de la siguiente manera: para cada $\mathbf{z}^1(k) = \left(\mathbf{x}^1(k), \mathbf{d}^1(k)\right)$ el modelo GP de la dinámica es usado para estimar la media de los estados sucesores que componen el conjunto $\overline{\mathfrak{X}}_{k+1}^S$ a partir del cual los ϑ estados más promisorios, de acuerdo a la función de utilidad (16), son elegidos como los mejores que formarán parte del conjunto $\mathfrak{X}_k^S$.

Una vez que la recursión ha sido completada, una primera versión de la política de conmutación $\pi_1^*(\mathfrak{X}_k^S), k = 1, \dots, N$ es aproximada usando un proceso Gaussiano. En la siguiente iteración, la política π_1^* es aplicada a todas las entradas iniciales $\mathbf{z}^2(0) \in \mathfrak{X}_0^S$ de modo que se obtienen tuplas $\left(\mathbf{x}^2(k), \mathbf{d}^2(k), \boldsymbol{\theta}^2(k)\right)$ a lo largo de las trayectorias simuladas de manera que nuevos pares de estados/perturbaciones pueden ser considerados como candidatos para aumentar $\mathfrak{X}_k^S, k = 1, \dots, N$. El aprendizaje Bayesiano activo es entonces usado para seleccionar los ϑ estados más informativos y el modelo $\mathcal{GP}_f$ que modela la dinámica de transición de estados/perturbación es actualizado. Cuando la recursión finaliza se obtiene una nueva versión de la política de conmutación $\pi_2^*(\mathfrak{X}_k^S), k = 1, \dots, N$.

El algoritmo BA$\mathcal{L}$GPDP tiene una diferencia radical con los algoritmos $\mathcal{L}$GPDP y $\mathcal{L}$DP debido a su enfoque iterativo para mejorar la política. Puesto que la función de valor $V_k^*(\mathfrak{X}_k^S),\ k = 1, \dots, N$ es aproximada desde la iteración dos en adelante. Pensar condiciones de terminación relacionadas con dicha función de valor puede resultar muy ventajoso para monitorear continuamente los estados visitados por el sistema como consecuencia de la ejecución de un determinado modo de control. Para este propósito, un diseño efectivo de una condición de terminación puede relacionarse con el gradiente de la función de valor, es decir, un modo de control se mantiene activo mientras que los valores de los estados encontrados, en promedio, sean monótonamente creciente. Tan pronto como los valores de los estados visitados comiencen a disminuir, una condición de terminación se debe disparar. Como resultado, el muestreo selectivo del EE a través de las condiciones de terminación basadas en la aproximación de la función de valor sobre $\mathfrak{X}$ es un elemento clave del procedimiento de iteración de la política, a fin de evitar visitar estados con valores extremadamente bajos y que prácticamente no proporcionen información relevante.

La *p-ésima* política mejorada es modelada como $\pi_p \sim \mathcal{GP}_\pi(m_\pi, Cov_\pi)$. Para determinar la necesidad de una nueva

iteración se requiere definir un criterio de paro (línea 30), por ejemplo, en base a las distancias de *Kullback-Leibler* (KL) (Verdinelli y Kadane 1992) de las políticas de control halladas en las sucesivas iteraciones. Utilizando estas magnitudes, es posible proponer diferentes variantes como criterio de paro del algoritmo. En la medida que en las sucesivas iteraciones el proceso de aprendizaje avanza hacia la política óptima, las distancias de KL entre las políticas sucesivas convergen a un determinado valor $\mathbf{e}$ menor que un umbral de tolerancia δ, tal que:

$$\begin{aligned}\Re_{previous} &= \sum_{\mathbf{x}\in\mathfrak{X}} KL\left(\mathcal{GP}_{\pi^*_{p-2}(\mathfrak{X})}\middle\|\mathcal{GP}_{\pi^*_{p-2}(\mathfrak{X})}\right)\Big/\|\mathfrak{X}\| = \\ &= \sum_{\mathbf{x}\in\mathfrak{X}} \mathcal{GP}_{\pi^*_{p-1}(\mathbf{x})}.\log\left(\mathcal{GP}_{\pi^*_{p-1}(\mathbf{x})}\Big/\mathcal{GP}_{\pi^*_{p-2}(\mathbf{x})}\right)\Big/\|\mathfrak{X}\| \\ \Re_{current} &= \sum_{\mathbf{x}\in\mathfrak{X}} KL\left(\mathcal{GP}_{\pi^*_{p-1}(\mathfrak{X})}\middle\|\mathcal{GP}_{\pi^*_{p}(\mathfrak{X})}\right)\Big/\|\mathfrak{X}\| \\ &= \sum_{\mathbf{x}\in\mathfrak{X}} \mathcal{GP}_{\pi^*_{p}(\mathbf{x})}.\log\left(\mathcal{GP}_{\pi^*_{p}(\mathbf{x})}\Big/\mathcal{GP}_{\pi^*_{p-1}(\mathbf{x})}\right)\Big/\|\mathfrak{X}\|\end{aligned} \quad (17)$$

5.2. Ejemplo de guiado del barco (continuación)

Nuevamente recurrimos al ejemplo del guiado de la embarcación tratado anteriormente para probar el desempeño de las propuestas. Particularmente, en este caso no se requiere que los modos sean totalmente fijados y parametrizados con antelación, por lo que se propone una ley de control igual para todos los modos pero sus parámetros son libres y deben ser aprendidos y optimizados a medida que progresa el entrenamiento. Los parámetros de modo pueden tomar valores en un dominio continuo, lo que da lugar a un continuo de opciones de modos de control. Además, las condiciones de terminación no solo dependen de la ocurrencia de eventos exógenos (que en el caso del ejemplo es atravesar los ejes medios $\mathsf{x}=0$, $\mathsf{y}=0$ ó salirse de las fronteras delimitadas) sino que también se las hace depender de la ocurrencia de eventos endógenos los cuales se derivan de la dirección del gradiente de la función de valor óptima.

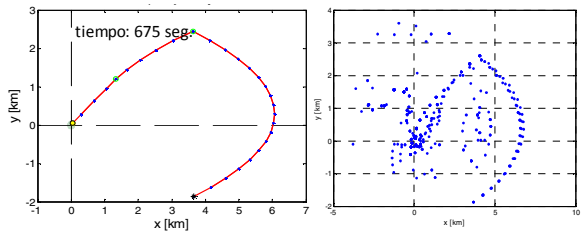


Figura 4. Resultado de aplicar BA$\mathcal{L}$GPDP al ejemplo determinístico del barco dirigido. a)Trayectoria del barco. b) Espacio de estado explorado.

En la Fig. 4 se ven los resultados de aplicar el Algoritmo 5 - BA$\mathcal{L}$GPDP- para resolver el ejemplo planteado en la Sección 4.1. De la comparación de las Fig. 3 y Fig. 4, las soluciones son muy parecidas, pero se advierte que la encontrada por el Algoritmo 5 es ligeramente mejor que la encontrada por el Algoritmo 2, para la versión determinista del problema. Ahora, cuando aparecen condiciones de incertidumbre, no es posible aplicar el Algoritmo 2, por lo que debemos recurrir a otra alternativa, como por ejemplo a la formulación realizada en el Algoritmo 5.

5.3. Ejemplo de guiado del barco no determinista

Seguidamente, se considera el problema de barco dirigido en un entorno incierto, donde por ejemplo el perfil de corriente de agua no sigue un patrón determinístico. Este cambio en la forma de la perturbación se realizó con un doble propósito. Por una parte, para modelar un problema más cercano a la realidad que se enfrenta en el control de embarcaciones en ambientes con perturbaciones prácticamente desconocidas. Por otra parte, para dejar en evidencia que el algoritmo propuesto en este trabajo es capaz de abordar este tipo de situaciones, que son más cercanas a una aplicación de ingeniería, que no han sido consideradas en trabajos similares como los presentados en la Sección 2. Para simular el nuevo caso, se emplea el siguiente modelo:

$$\begin{aligned}\dot{\mathsf{x}} &= C\cos(\phi) - \mathsf{y}\cdot C\cdot(1+N_x) \\ \dot{\mathsf{y}} &= C\sin(\phi) + C\cdot N_y\end{aligned} \quad (18)$$

dónde N_x es un valor que se define de acuerdo a una distribución normal con media 0 y desviación estándar que depende de la coordenada y; esto es: $N_\mathsf{x} = \mathcal{N}(0, |y|/10)$. El valor N_y se define como un valor con media 0 y desviación estándar C; esto es: $N_\mathsf{y} = \mathcal{N}(0, C)$.

Como puede verse en el sistema de ecuaciones (18) la dinámica posee términos aleatorios que a su vez dependen de la posición del barco en el plano coordenado x,y. Resolver este problema mediante una formulación analítica de un problema de control óptimo puede ser extremadamente complejo. Entonces, aquí se recurre a la propuesta realizada a lo largo de este trabajo, para lo cual es posible aplicar el algoritmo general BA$\mathcal{L}$GPDP de la Tabla 5 para resolver dicho problema.

En este caso, el algoritmo BA$\mathcal{L}$GPDP es aplicado al sistema usando como entrada un conjunto finito de parámetros de modo de control $\phi \in \Theta^S$ discretizado uniformemente en 15 valores sobre el intervalo $\phi \in [0, 2\pi]$. Además como política inicial π_0 bien se podría tomar una política aleatoria, pero se considera la política multimodal hallada en la Sección 2.4, la cual posee un pobre desempeño.

Como resultado, luego de 25 iteraciones, con un parámetro de descuento $\gamma = 0.95$ y un número máximo de etapas $N = 50$, el conjunto de soporte $\mathfrak{X}$ resultante para los estados relevantes proporciona suficiente información para aproximar la política de conmutación usando GPs. En la Fig. 5 se indican las trayectorias resultantes de aplicar las políticas de conmutación π^*_p, $p = 22, \ldots, 25$ obtenidas en cada una de las últimas 4 iteraciones del algoritmo BA$\mathcal{L}$GPDP. El objetivo de esta figura es mostrar como incrementalmente el algoritmo va mejorando las soluciones, hasta alcanzar una solución óptima, $\pi^* = \pi^*_{25}$, que da como resultado una trayectoria que insume tan solo 700 segundos para guiar al barco desde el punto de partida hasta la meta.

Como puede observarse en cada uno de los paneles de la Fig. 5, es posible advertir que se encuentra una buena solución, donde una de las trayectorias alcanzadas bajo la política óptima insume tan solo 700 segundos. Estas trayectorias, ponen de manifiesto que las políticas halladas por el algoritmo son cada vez mejores para manipular el sistema estocástico. El último panel (inferior derecho) de la Fig. 5 revela que la política de control óptima encontrada aprende a beneficiarse de los efectos de la corriente de agua para guiar el barco, desde la partida hacia la meta, mejorando notablemente el tiempo insumido.

Con el fin de contrastar la solución alcanzada por el Algoritmo 5 para el problema no determinístico, se empleó el Algoritmo 1 para resolver este mismo problema. La configuración de modos y parámetros empleada para el Algoritmo 1, en este caso, es idéntica a la expuesta en la Sección 3.3. En la Fig. 6 se indican los resultados alcanzados para 10^6 repeticiones aplicando el Algoritmo 1. Es notorio que los resultados obtenidos con el

Algoritmo 1 son significativamente inferiores a los alcanzados por el Algoritmo 5.

Seguidamente, con el fin de comparar el desempeño de los algoritmos 1 y 5 se usaron las políticas obtenidas por cada uno ellos para realizar pruebas independientes de guiado de la embarcación en el entorno no determinista. Denotamos como π^*_{Alg1} a la política óptima obtenida por el Algoritmo 1 y π^*_{Alg5} a la obtenida por el Algoritmo 5. La Fig. 7 muestra los resultados de 50 pruebas independientes para cada caso, resultantes de controlar el sistema mediante π^*_{Alg1} y π^*_{Alg5}. En esta Fig. 7 el desempeño de cada una de las políticas halladas se considera en términos del tiempo total transcurrido en guiar la embarcación desde la meta hacia el destino, en el entorno no determinísitico. Como puede observarse, la política π^*_{Alg5} consigue siempre mejores resultados que π^*_{Alg1}. Como resultado, el tiempo medio de viaje sobre las 50 pruebas con π^*_{Alg5} es de 758.5 segundos mientras que sobre las 50 pruebas con π^*_{Alg1} es de 1182.5 segundos.

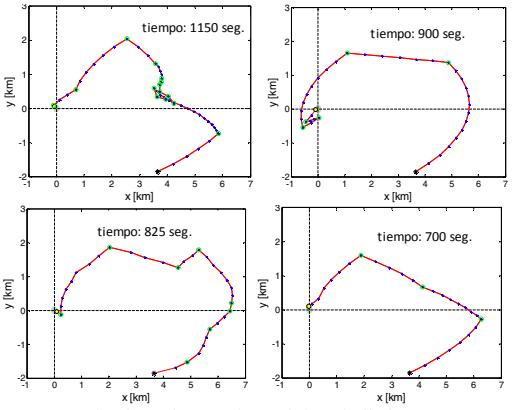


Figura 5. Trayectorias logradas por las políticas halladas con BA$\mathcal{L}$GPDP en las últimas 4 iteraciones para el caso no determinista.

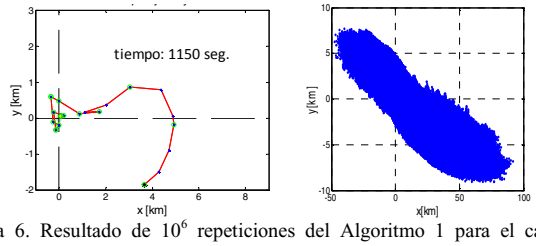


Figura 6. Resultado de 10^6 repeticiones del Algoritmo 1 para el caso no determinista. a) Trayectoria del barco. b) Espacio de estado explorado

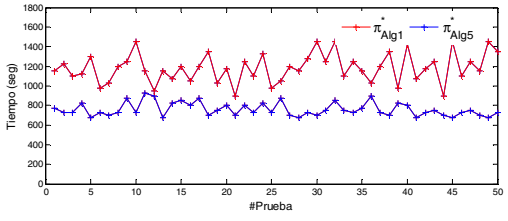


Figura 7. Resumen de 50 pruebas independientes de las políticas óptimas π^*_{Alg1} y π^*_{Alg5}, respectivamente.

En la Fig.8, se muestra una gráfica donde se encuentran los valores de función de los estados que forman el conjunto de datos de soporte de la política óptima hallada por el Algoritmo 5. En este ejemplo es interesante ver el efecto de emplear el mecanismo de aprendizaje Bayesiano activo, a través del cual se introduce un sesgo para encontrar aquellos estados más promisorios, informativamente hablando. Este efecto puede verse claramente en dicha figura, donde los estados más cercanos a la meta son los que poseen mayor valor y a medida que se aleja de la zona objetivo se observa que las estimaciones de la función de valor, sobre estos puntos más alejados, van reduciendo su valor. Mediante este mecanismo, se logra que el algoritmo pueda arribar más rápidamente a una solución estimando los valores de estados en una región de interés para alcanzar la meta, evitando explorar regiones no deseadas o irrelevantes.

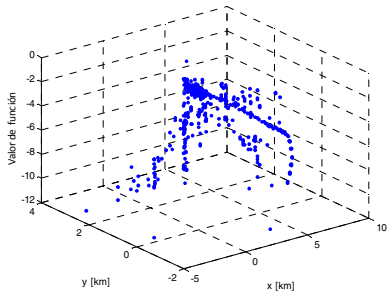


Figura 8. Valores de función de los estados del conjunto de soporte de π^*.

6. Conclusiones

El eje central de la investigación realizada aborda el desarrollo de estrategias de control multimodal para sistemas en entornos inciertos. Tres cuestiones importantes han sido tratadas en el desarrollo de los algoritmos de PD para un control de conmutación cuasi óptimo en presencia de incertidumbre. La primera cuestión se refiere a la eficiencia del muestreo de un EE continuo en PD usando un número finito de modos de control. El enfoque adoptado fue motivado por el reemplazo del muestreo convencional de Riemann por el muestreo de Lebesgue en los algoritmos de la PD clásica. Para muestrear el EE mediante la concatenación de modos de control en secuencias de diferentes longitudes, fue propuesto el Algoritmo 3 (Tabla 3) e integrado con PD para dar lugar al Algoritmo 2 (Tabla 2) oportunamente llamado $\mathcal{L}$DP, o Programación Dinámica basada en muestreo de Lebesgue. La segunda cuestión está relacionada con el hecho de lidiar con incertidumbre en las transiciones de estados y la necesidad de generalización de las funciones de valor. Debido al efecto variable de las acciones de control aparecen estados desconocidos aún cuando es aplicada la misma secuencia de modos de control. Desde que la política de conmutación proporcionada por $\mathcal{L}$DP define los modos de control óptimos sólo para los estados generados con el Algoritmo 3, fue incorporada la capacidad de generalización usando procesos Gaussianos lo que dio lugar al novedoso algoritmo BA$\mathcal{L}$GPDP para control multimodal. Se abordó también la cuestión relacionada con la introducción de un sesgo adecuado en los datos, para hacer más eficientes el muestreo de Lebesgue y el algoritmo $\mathcal{L}$GPDP, mediante el uso de un mecanismo de aprendizaje Bayesiano activo en los experimentos de evaluación de la política. Así, con los datos seleccionados y recolectados es posible aprender un *metamodelo* de la dinámica de transición de modos, $\mathcal{GP}_f$, lo cual es el instrumento que hace al algoritmo propuesto BA$\mathcal{L}$GPDP ideal para aprender, a través de una simulación intensiva, una

política de conmutación y luego adaptarla a las verdaderas condiciones del entorno operativo del sistema dinámico. Un caso de estudio representativo fue presentado para realizar una serie de ejemplos numéricos que ayudan a ilustrar la aplicabilidad de las propuestas realizadas.

English Summary

Multimodal Control in Uncertain Environments using Reinforcement Learning and Gaussian Processes.

Abstract

The control of complex systems can be done decomposing the control task into a sequence of control modes, or modes for short. Each mode implements a parameterized feedback law until a termination condition is activated in response to the occurrence of an exogenous/endogenous event, which indicates that the execution mode must end. This paper presents a novel approach to find an optimal switching policy to solve a control problem by optimizing some measure of cost/benefit. An optimal policy implements an optimal multimodal control program, consisting in a sequence of control modes. The proposal includes the development of an algorithm based on the idea of dynamic programming integrating Gaussian processes and Bayesian active learning. In addition, an efficient use of the data to improve the exploration of the continuous state spaces solutions can be achieved through this approach. A representative case study is discussed and analyzed to demonstrate the performance of the proposed algorithm.

Keywords:

Multimodal Control, Dynamic Programming, Gaussian Processes, Uncertainty, Policy.

Referencias

Abate, Alessandro, Maria Prandini, John Lygeros, Shankar Sastry. 2008. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. Automatica 44, 2724-2734.

Adamek, F., M Sobotka, O Stursberg. 2008. Stochastic optimal control for hybrid systems with uncertain discrete dynamics. Proceedings of the IEEE International Conference on Automation Science and Engineering, 23-28. Washington D.C.

Åström, Karl Johan, Bo Bernhardsson. 2003. System with Lebesgue Sampling. Directions in Mathematical Systems Theory and Optimization, LNCIS 268. LNCIS. Springer-Verlag Berlin Heidelberg.

Axelsson, H., Y. Wardi, M. Egerstedt, E. I. Verriest. 2007. Gradient descent approach to optimal mode scheduling in hybrid dynamical systems. Journal of Optimization Theory and Applications 136, 167-186.

Azuma, Shun-ichi, Jun-ichi Imura, Toshiharu Sugie. 2010. Lebesgue piecewise affine approximation of nonlinear systems. Nonlinear Analysis: Hybrid Systems 4, 92-102.

Barton, Paul I., Cha Kun Lee,Mehmet Yunt. 2006. Optimization of hybrid systems. Computers & Chemical Engineering 30, 1576-1589.

Bemporad, A.,S. Di Cairano. 2011. Model-predictive control of discrete hybrid stochastic automata. Automatic Control, IEEE Transactions on 56, 1307 -1321.

Bemporad, Alberto,Manfred Morari. 1999. Control of systems integrating logic, dynamics, and constraints. Automatica 35, 407-427.

Bensoussan, A.,J. L. Menaldi. 2000. Stochastic hybrid control. Journal of Mathematical Analysis and Applications 249.

Bertsekas, Dimitri P. 2000. Dynamic Programming and Optimal Control, Vol. I. 2nd ed. Athena Scientific.

Blackmore, L., M. Ono, A. Bektassov,B.C. Williams. 2010. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. Robotics, IEEE Transactions on 26, 502-517.

Borrelli, Francesco, Mato Baotić, Alberto Bemporad,Manfred Morari. 2005. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. Automatica 41, 1709-1721.

Bryson, Jr Arthur E.,Yu-Chi Ho. 1975. Applied optimal control: optimization, estimation and control. Revised. Taylor & Francis.

Busoniu, Lucian, Robert Babuska, Bart De Schutter,Damien Ernst. 2010. Reinforcement learning and dynamic programming using function approximators. 1.a ed. CRC Press.

Cassandras, Christos G.,John Lygeros. 2007. Stochastic hybrid systems. Boca Raton: Taylor & Francis.

Deisenroth, Marc Peter. 2010. Efficient Reinforcement Learning Using Gaussian Processes. KIT Scientific Publishing.

Deisenroth, Marc Peter, Carl Edward Rasmussen,Jan Peters. 2009. Gaussian process dynamic programming. Neurocomputing 72, 1508-1524.

Di Cairano, S., A. Bemporad,J. Júlvez. 2009. Event-driven optimization-based control of hybrid systems with integral continuous-time dynamics. Automatica 45, 1243-1251.

Ding, X.-C., Y. Wardi,M. Egerstedt. 2009. On-line optimization of switched-mode dynamical systems. Automatic Control, IEEE Transactions on 54, 2266 -2271.

Egerstedt, M., Y. Wardi,H. Axelsson. 2006. Transition-Time Optimization for Switched-Mode Dynamical Systems. IEEE Transactions on Automatic Control 51, 110- 115.

Girard, Agathe. 2004. Approximate methods for propagation of uncertainty with gaussian process models. University of Glasgow.

Kuss, M. 2006. Gaussian process models for robust regression, classification, and reinforcement learning. Technische Universite Darmstadt.

Liberzon, Daniel. 2003. Switching in systems and control. Systems & Control: Foundations & Applications. Boston: Birkhäuser Boston Inc.

Lincoln, B.,A. Rantzer. 2006. Relaxing Dynamic Programming. IEEE Transactions on Automatic Control 51, 1249-1260.

Lunze, Jan,Daniel Lehmann. 2010. A state-feedback approach to event-based control. Automatica 46, 211-215.

Mehta, Tejas,Magnus Egerstedt. 2005. Learning multi-modal control programs. Hybrid Systems: Computation and Control, 466-479. Lecture Notes in Computer Science. Springer Berlin

Mehta, Tejas R.,Magnus Egerstedt. 2006. An optimal control approach to mode generation in hybrid systems. Nonlinear Analysis 65, 963-983.

Mehta, Tejas R., Magnus Egerstedt. 2008, Multi-modal control using adaptive motion description languages. Automatica 44, 1912-1917.

Pajares Martin-Sanz, G., y De la Cruz Garcia J.M. 2010. Aprendizaje automático. Un enfoque práctico, Cap. 12, Aprendizaje por Refuerzos. RA-MA.

Rantzer, A. 2006. Relaxed Dynamic Programming in Switching Systems. Control Theory and Applications, IEE Proceedings - 153, 567- 574.

Rasmussen, Carl Edward,Christopher K. I. Williams. 2006. Gaussian processes for machine learning. MIT Press.

Rosenstein, Michael T.,Andrew G. Barto. 2004. Supervised Actor-Critic Reinforcement Learning. Handbook of Learning and Approximate Dynamic Programming, 359–380. John Wiley & Sons, Inc.

Salichs, M. A., Malfaz, M., y Gorostiza, J.F. 2010. Toma de Decisiones en Robótica. Revista Iberoamericana de Automática e Informática Industrial RIAI 7, 5-16.

Shi, Peng, Yuanqing Xia, G.P. Liu,D. Rees. 2006. On designing of sliding-mode control for stochastic jump systems. Automatic Control, IEEE Transactions on 51, 97 - 103.

Song, Chunyue,Ping Li. 2010. Near optimal control for a class of stochastic hybrid systems. Automatica 46, 1553-1557.

Sutton, Richard S.,Andrew G. Barto. 1998. Reinforcement learning: An introduction. MIT Press.

Verdinelli, Isabella,Joseph B. Kadane. 1992. Bayesian designs for maximizing information and outcome. Journal of the American Statistical Association 87, 510-515.

Xu, Xuping,Panos J. Antsaklis. 2003. Results and perspectives on computational methods for optimal control of switched systems. Proceedings of the 6th international conference on Hybrid systems: computation and control, 540–555. Springer-Verlag.

Xu, Yan-Kai,Xi-Ren Cao. 2011. Lebesgue-Sampling-Based Optimal Control Problems with Time Aggregation. Automatic Control, IEEE Transactions on 56, 1097-1109.

Zhang, Wei, Jianghai Hu,A. Abate. 2009. On the value functions of the discrete-time switched LQR problem. Automatic Control, IEEE Transactions on 54, 2669 -2674.