Algoritmo SCDO en el Principio de Mínimo y la programación dinámica definida sobre dominios finitos

The Minimum Principle on SCDO algorithm and the dynamic programming defined over finite domaine

J. Cardillo* y F. Szigeti
Departamento de Sistemas de Control, Facultad de Ingeniería, ULA,
Mérida 5101, Venezuela
*ijuan@ing.ula.ve

J-C. Hennet y J-L Calvet
LAAS-CNRS, Groupe MOSIGA, Toulouse France, Cedex 4 31000

Resumen

En este articulo se muestra un algoritmo basado en cálculo formal, el cual obtiene la forma explicita del mínimo sobre funciones definidas sobre dominios finitos. Este algoritmo es usado para obtener las expresiones del control óptimo (mínimo) cuando se usan los métodos del principio de mínimo para procesos sobre dominios finitos y la programación dinámica paramétrica.

Palabras Claves: Cálculo formal, programación dinámica, principio de mínimo, sistemas a eventos discretos, mínimos de funciones sobre dominios finitos.

Abstract

In this paper we show an algorithm based on formal calculus, which obtains the explicit form of the minimum on functions defined over finite domain. This algorithm is used to obtain expressions of the optimal control (minimum) when the methods: the principle of minimum for processes on finite domain and the parametric dynamic programming are used.

Keywords: Formal calculus, dynamic programming, minimum principle, discrete event dynamical systems, minimum of functions defined over finite domains.

1 Introducción

El uso de principios en la teoría de optimización clásica, genera condiciones necesarias, suficientes o ambas, de tal manera que al aplicar algún método, algoritmo, etc. es posible conseguir que para ciertos valores del dominio, las condiciones se satisfacen, así, en principio, se proporciona la solución al problema planteado. Estas condiciones de optimalidad, en la mayoría de los casos cuando el problema planteado está descrito sobre dominios discretos finitos se traducen en obtener mínimos de funciones definidas sobre dominios discretos. La optimización de funciones definidas sobre un conjunto discreto es generalmente realizada puramente por algoritmos enumerativos numéricos, con severas limitaciones en el número de variables. Como un enfoque alternativo, aquí proponemos el uso de la computación simbólica para obtener iterativamente la solución óptima formal de un problema de minimización sobre un conjunto finito de variables Booleanas.

Una fórmula explícita de la optimización de una variable es primero obtenida, usando como la base del dominio Booleano de una variable al conjunto {-1,1}, en vez del conjunto {0,1}. Entonces el cálculo simbólico provee fórmulas explícitas para el caso Booleano multivariable. Los dominios finitos de variables enteras pueden ser transformados en un dominio Booleano finito con la ayuda de una

representación diádica de enteros.

Adicionalmente, el polinomio de interpolación de Lagrange provee una representación general de funciones definida sobre conjuntos discretos. Una combinación de estas herramientas genera una representación canónica de funciones definidas sobre conjuntos discretos. La aplicación de esta representación canónica de funciones es caracterizada por una secuencia de matrices A_k, las cuales exhiben algunas propiedades que se resaltan en este artículo. El uso de estas propiedades reduce considerablemente la complejidad computacional del método. Este método alternativo con lleva a un algoritmo el cual hemos denominado SCDO. Debido a la naturaleza simbólica del algoritmo SCDO es posible incorporar parámetros exógenos a las funciones a minimizar.

Así, una aplicación directa del algoritmo SCDO, esta dada en la obtención de una formula explicita para el control óptimo cuando:

- el Principio de Mínimo para procesos a eventos discretos sobre dominios finitos es usado.
- la programación dinámica sobre dominios finitos es usada.

En ambos casos parámetros exógenos pueden aparecer tanto en las restricciones como en la función de costo.

2 Optimización simbólica de funciones booleanas

Aquí, el mínimo absoluto de funciones Booleanas será expresado por una fórmula explícita. El uso de esta fórmula es propuesta como una alternativa a los métodos de búsqueda clásicos. Una particular característica del enfoque propuesto es el uso de valores Booleanos en el conjunto {-1,1} en vez del conjunto {0,1} como dominio binario unidimensional básico.

Así, el dominio de la función puede escogerse como un subconjunto de los vértices del hipercubo dado por:

$$D \subset \{-1,1\}^{k+1},\ f{:}D \to \Re.$$

Por conveniencia, una función definida sobre D puede ser extendida al conjunto entero del producto $\{-1,1\}^{k+1}$, de la siguiente manera. Fijando un $x_0 \in D$ arbitrario, una apropiada extensión de f sobre $\{-1,1\}^{k+1}$ es dada por:

$$\bar{f}_{x_0} : \times\{-1,1\} \to \Re,$$
$$\bar{f}_{x_0}(x) = \begin{cases} f(x) & \text{if } x \in D, \\ f(x_0)+1 & \text{if } x \in \{-1,1\}^{k+1} \backslash D. \end{cases} \quad (1)$$

Es claro que el mínimo absoluto y los puntos donde es alcanzado, son los mismos para la función f y $\bar{f}_{x_0}$, aquí podemos suponer, sin perder generalidad, que el dominio de definición de la función Booleana es el conjunto producto: $\{-1,1\}^{k+1}$.

2.1. Caso monovariable

Consideremos, $f : \{-1,1\} \to \Re$. Entonces el lemma siguiente puede ser enunciado:

Lema 2.1:

El valor mínimo de la función Booleana f es obtenido en

$$u^* = -\text{Sign}[f(1) - f(-1)] = \text{Sign}[f(-1) - f(1)], \quad (2)$$

donde la función Sign es usada en el sentido siguiente:

$$\text{Sign}(t) = \begin{cases} 1 & \text{if } t>0, \\ -1 & \text{if } t<0, \end{cases} \quad (3)$$

y, para t=0, Sign(0)= {-1,1} es un conjunto valuado. En el último caso, la función f obtiene su óptimo (mínimo o máximo) en ambos puntos del dominio.

La prueba de este lema es obtenida de forma inmediata, considerando los dos posible valores $u^* = -1$ ó $u^* = 1$.

2.2 Caso multivariable

La aplicación del Lema 2.1 para el caso vectorial toma la forma de un algoritmo el cual contiene dos partes. La primera parte es un cálculo simbólico hacia atrás, que consiste de k+1 pasos como se describe abajo.

En cada paso, el Lema 2.1 es usado para proveer una expresión simbólica del valor óptimo $u_i^+(u_0,\cdots,u_{i-1})$. La segunda parte, comienza con el cálculo explícito del valor óptimo de u_0^*. Entonces se procede hacia delante para calcular la secuencia óptima de los u_i^*, para i=1, …, k.

Las dos partes del algoritmo serán mostradas con detalle a continuación:

Parte 1

Considere la función:

$$f_0(u_0,\ldots,u_{k-1},u_k) = f(u_0,\ldots,u_{k-1},u_k).$$

- **Paso 1**: Se define la función parcial:

$$u_k \to f_0(u_0,\ldots,u_k) \text{ para } u_0,\ldots,u_{k-1} \text{ fijos.}$$

Por el Lema 2.1, el mínimo del problema de optimización paramétrica es obtenido como:

$$u_k^+(u_0, u_1, ..., u_{k-1}) = \text{Sign}(f_0(u_0, u_1, ..., u_{k-1}, -1) - f_0(u_0, u_1, ..., u_{k-1}, 1))$$

Se define la función $f_1 : \{-1,1\}^k \to \Re$, como:

$$f_1(u_0, ..., u_{k-1}) = f_0(u_0, ..., u_{k-1}, u_k^+(u_0, ..., u_{k-1}))\ .$$

- **Paso i:** Suponga que las funciones:

$$f_2 : \{-1,1\}^{k-1} \to \Re, \cdots, f_{i-1} : \times_0^{k-i+1}\{-1,1\} \to \Re,$$

están generadas como en el paso 1 y sus óptimos simbólicos son alcanzados en:

$$u_{k-1}^+(u_0, \ldots, u_{k-2}), \cdots, u_{k-i+2}^+(u_0, \ldots, u_{k-i+1})\ .$$

Entonces, a partir del Lema 2.1, la función parcial:

$$u_{k-i+1} \to f_{i-1}(u_0, \ldots, u_{k-i}, u_{k-i+1})\ ,$$

alcanza su mínimo en:

$$u_{k-i}^+(u_0, u_1, ..., u_{k-i-1}) = \text{Sign}(f_i(u_0, u_1, ..., u_{k-i-1}, -1) - f_i(u_0, u_1, ..., u_{k-i-1}, 1)),$$

y la función a ser optimizada puede ser reemplazada por la función $f_{i+1} : \{-1,1\}^{k-i+1} \to \Re$, definida por:

$$f_{i+1}(u_0, ..., u_{k-i-1}) = f_i(u_0, ..., u_{k-i-1}, u_{k-i}^+(u_0, ..., u_{k-i-1}))\ .$$

- **Paso k:** La última función:

$f_k : \{-1,1\} \to \Re$, alcanza su valor mínimo en :

$$u_0^* = u_0^+ = \text{Sign}\left(f_k(-1) - f_k(1)\right),$$

el cual es un subconjunto {-1,1}. Por el Lema 2.1, nosotros conocemos que u_0^* pertenece al conjunto {-1,1} solamente en el caso de que f_k sea constante sobre {-1,1}.

Parte 2:

El cálculo de los valores óptimos, u_i^*, donde $f_{k-i}(.)$ alcanza el mínimo, es obtenida por substitución hacia delante de la siguiente manera:

- **Paso 1**: La condición inicial es el paso k de la parte 1.
- **Paso i**, para i=1,..,k. Suponga que u_0^*, u_1^*, …, u_{i-1}^* se han calculado. Entonces u_i^* puede ser calculado por la recursión:

$$u_i^* = u_i^+\left(u_0^*, \ldots, u_{i-1}^*\right) = \text{Sign}\left(f_{k-i}\left(u_0^*, \ldots, u_{i-1}^*, -1\right) - f_{k-i}\left(u_0^*, \ldots, u_{i-1}^*, 1\right)\right).$$

Dos posibles casos se pueden distinguir:

Caso 1:
Si $f_{k-i}\left(u_0^*, u_1^*, \ldots, u_{i-1}^*, -1\right) \neq f_{k-i}\left(u_0^*, u_1^*, \ldots, u_{i-1}^*, 1\right)$,
entonces $u_i^* \in \{-1,1\}$

Caso 2:
Si $f_{k-i}\left(u_0^*, u_1^*, \ldots, u_{i-1}^*, -1\right) = f_{k-i}\left(u_0^*, u_1^*, \ldots, u_{i-1}^*, 1\right)$,
entonces, los dos posibles valores, -1 o 1, son admisibles para u_{k-i}^*.

Este caso genera dos secuencias óptimas las cuales son exploradas en el siguiente paso.

Así el número de secuencias óptimas se incrementan en uno cada vez que el caso 2 ocurre.

3 Optimización sobre dominios finitos

Consideremos ahora una función real general sobre un dominio discreto finito $g(x) : D_g \subset Z^N \to \Re$. El caso multidimensional (N>1) puede ser transformado en el caso monovariable por enumeración.

El caso de dominio finito es trasladado en el previamente definido dominio Booleano usando el siguiente lema.

Lema 2.2:

Considere el conjunto:

$\{0,1,\ldots,n\}$, y $\{-1,1\}^{k+1}$

Si k es el entero más pequeño tal que $n < 2^{k+1}$, entonces la aplicación:

$$\psi : \{0,1,\ldots,n\} \to \{-1,1\}^{k+1},$$

es unívocamente definida por su inversa:

$$u = \chi\left(u_0, \ldots, u_k\right) = \frac{1}{2}\left(1+u_0\right) + \left(1+u_1\right) + \cdots + 2^{k-1}\left(1+u_k\right).$$

La prueba del Lema 2.2 sigue de la unicidad de la representación diádica de algún $n \in N$, para $2^k - 1 < n \leq 2^{k+1} - 1$.

La aplicación del Lema 2.2 en el caso de $n = \text{Car}\left(D_g\right)$

permite transformar la función $g(x): D_g \to \Re$ en la función $\varphi(u_0,\ldots,u_k): \{-1,1\}^{k+1} \to \Re$ sobre la condición $2^k - 1 < n \le 2^{k+1} - 1$. La función $\varphi(u_0,\ldots,u_k)$ es entonces unívocamente definida por:

$$\varphi = \chi \circ g .$$

Usando la extensión descrita en la ecuación (1), puede suponerse, sin restricción, que el dominio de φ es $\{-1,1\}^{k+1}$.

3.1. Representación polinómica canónica

Cualquier función definida sobre un dominio finito puede reemplazarse, equivalentemente, por un polinomio que toma los mismos valores que la función original en cada punto discreto.

En particular, cualquier función Booleana $\varphi(u_0,\ldots,u_k)$ como la definida antes puede ser reformulada como una función polinómica $f(u_0,\ldots,u_k)$ sobre un dominio $D \subset \{-1,1\}^{k+1}$, como:

$$f(u_0,\ldots,u_k) = \sum_{\underline{\alpha}} c_{\underline{\alpha}} u_0^{\alpha_0} \ldots u_k^{\alpha_k} , \tag{4}$$

con un número finito de secuencias $\underline{\alpha} = (\alpha_0, \ldots, \alpha_k)$ que pertenezcan a un subconjunto finito de N^{k+1}.

En el caso de una función φ definida sobre un dominio $D \subset \{-1,1\}^{k+1}$, tenemos adicionalmente la siguiente propiedad:

$$\begin{aligned} &u_i^p = 1 \text{ si p es par} \\ &u_i^p = u_i \text{ si p es impar.} \end{aligned} \tag{5}$$

Por lo tanto en todos los monomios implicados en la representación (4), esto es: $u_0^{\alpha_0} \cdots u_k^{\alpha_k}$, es suficiente considerar $(\alpha_0, \ldots, \alpha_k) \in \{0,1\}^{k+1}$.

Tal representación también es provista por la interpolación de Lagrange sobre todos los puntos del dominio.

$$f(u_0,\ldots,u_k) = \sum_{\underline{\varepsilon} \in \{-1,1\}^{k+1}} \varphi(\varepsilon_0,\varepsilon_1,\cdots,\varepsilon_k) \frac{u_0+\varepsilon_0}{2\varepsilon_0} \frac{u_1+\varepsilon_1}{2\varepsilon_1} \cdots \frac{u_k+\varepsilon_k}{2\varepsilon_k} \tag{6}$$

$$f(\underline{\varepsilon}) = \varphi(\underline{\varepsilon}) .$$

En esta representación es obvio que todos los monomios en (6) tiene la forma $u_0^{\alpha_0}, u_1^{\alpha_1}, \cdots, u_k^{\alpha_k}$, donde $(\alpha_0, \alpha_1, \ldots, \alpha_k) \in \{0,1\}^{k+1}$. Aquí, todas las funciones sobre $\{-1,1\}^{k+1}$ pueden ser representadas como:

$$f(u_0,\ldots,u_k) = \sum_{(\alpha_0,\cdots,\alpha_k)\in\{0,1\}^{k+1}} f_{\alpha_0,\cdots,\alpha_k} u^{\alpha_0} \cdots u^{\alpha_k} . \tag{7}$$

Si hacemos la correspondencia entre los valores de la función con los vértices del hipercubo $\{-1,1\}^{k+1}$ en un vector de dimensión 2^{k+1}, denotado por $\tilde{F}_k$, y ordenamos los vértices del hipercubo en orden lexicográfico tenemos:

$$\tilde{F}_k = \left(f(-1,\cdots,-1,-1), f(-1,\cdots,-1,1), \cdots, f(1,\cdots,1,1)\right)^T .$$

Ahora, ordenamos los coeficientes de la expresión polinómica (7) en un vector de dimensión 2^{k+1}, en orden antilexicográfico , denotado por F_k, como:

$$F_k = \left(f_{111\cdots1}, \cdots, f_{100\cdots0}, f_{011\cdots1}, \cdots, f_{000\cdots0}\right)^T .$$

Entonces el siguiente lema puede ser enunciado.

Lema 2.3:

$$\begin{aligned} &\tilde{F}_i = A_i F_i \text{, para i=0,…,k con} \\ &A_0 = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \in \Re^{2\times 2}, \text{ y para i=1,}\cdots\text{,k,} \\ &A_i = \begin{bmatrix} -A_{i-1} & A_{i-1} \\ A_{i-1} & A_{i-1} \end{bmatrix} \in \Re^{2^{i+1}\times 2^{i+1}} . \end{aligned} \tag{8}$$

El Lema 2.3 puede ser probado por inducción matemática.

Para i=0, una función de una variable $u_0 \in \{-1,1\}$ puede ser escrita:

$$f(u_0) = f_1 u_0 + f_0, \; u_0 \in \{-1,1\} .$$

La relación entre los coeficientes f_0, f_1 y los valores f(-1), f(1) es dada por la siguiente ecuación lineal:

$$\begin{bmatrix} f(-1) \\ f(1) \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_0 \end{bmatrix}, \text{ esto es, } \tilde{F}_0 = A_0 F_0 .$$

Ahora, suponemos que para alguna función de i+1 variables binarias:

$$\tilde{F}_i = A_i F_i ,$$

entonces, para alguna función $f(u_0,\cdots,u_i,u_{i+1})$, con $(u_0,\cdots,u_i,u_{i+1})\in\{-1,1\}^{i+2}$,

$$f(u_0,\cdots,u_i,-1)=-\sum f_{\varepsilon_0\cdots\varepsilon_i 1}u_0^{\varepsilon_0}\cdots u_i^{\varepsilon_i}+\sum f_{\varepsilon_0\cdots\varepsilon_i 0}u_0^{\varepsilon_0}\cdots u_i^{\varepsilon_i},$$

conduce a:

$$\tilde{F}_{i+1,-1}=-A_iF_{i+1,1}+A_iF_{i+1,0} \tag{9}$$

y

$$f(u_0,\cdots,u_i,1)=\sum f_{\varepsilon_0\cdots\varepsilon_i 1}u_0^{\varepsilon_0}\cdots u_i^{\varepsilon_i}+\sum f_{\varepsilon_0\cdots\varepsilon_i 0}u_0^{\varepsilon_0}\cdots u_i^{\varepsilon_i},$$

conduce a:

$$\tilde{F}_{i+1,1}=A_iF_{i+1,1}+A_iF_{i+1,0}, \tag{10}$$

donde:

$$\tilde{F}_{i+1}=\begin{bmatrix}\tilde{F}_{i+1,-1}\\ \tilde{F}_{i+1,1}\end{bmatrix}\ \text{y}\ F_{i+1}=\begin{bmatrix}F_{i+1,1}\\ F_{i+1,0}\end{bmatrix}.$$

Así:

$$\tilde{F}_{i+1}=\begin{bmatrix}-A_i & A_i\\ A_i & A_i\end{bmatrix}\begin{bmatrix}F_{i+1,1}\\ F_{i+1,0}\end{bmatrix}=A_{i+1}F_{i+1},$$

con:

$$A_{i+1}=\begin{bmatrix}-A_i & A_i\\ A_i & A_i\end{bmatrix}\in\Re^{2^{i+2}\times 2^{i+2}}$$

En la práctica, los parámetros desconocidos son los componentes del vector F_i, para $i=1,\cdots,k$. Una de las principales ventajas de la representación canónica propuesta, es la simplicidad para invertir la matriz A_i, tal como lo indica el siguiente lema.

Previo al anuncio del lema daremos la fórmula de Shur para inversión de matrices particionadas. Sea $F\in M_{m+n\times m+n}$ una matriz particionada en bloques de la siguiente forma:

$$F=\begin{bmatrix}A & B\\ C & D\end{bmatrix}$$

con $A\in M_{m\times m}$ y $D\in M_{n\times n}$ ambas invertibles entonces :

$$F^{-1}=\begin{bmatrix}(A-BD^{-1}C)^{-1} & -A^{-1}B(D-CA^{-1}B)^{-1}\\ -D^{-1}C(A-BD^{-1}C)^{-1} & (D-CA^{-1}B)^{-1}\end{bmatrix}.$$

Lema 2.4:

La matriz A_i es invertible, y $A_i^{-1}=\frac{1}{2^{i+1}}A_i$.

Prueba

El Lema 2.4 también es probado por inducción matemática.

Para i=0, A_0 es dado por:

$$A_0=\begin{bmatrix}-1 & 1\\ 1 & 1\end{bmatrix},$$

entonces, obviamente:

$$A_0^{-1}=\frac{-1}{2}\begin{bmatrix}1 & -1\\ -1 & -1\end{bmatrix}=\frac{1}{2}A_0.$$

Supongamos ahora que A_i satisface lo dicho en el Lema 2.4. Usando la conocida fórmula de inversión de Shur para bloques de matrices particionadas, obtenemos:

$$A_{i+1}^{-1}=\begin{bmatrix}-A_i & A_i\\ A_i & A_i\end{bmatrix}^{-1}=\begin{bmatrix}-\frac{1}{2}[A_i]^{-1} & \frac{1}{2}[A_i]^{-1}\\ \frac{1}{2}[A_i]^{-1} & \frac{1}{2}[A_i]^{-1}\end{bmatrix}.$$

Así la hipótesis inductiva muestra:

$$A_{i+1}^{-1}=\frac{1}{2^{i+2}}\begin{bmatrix}-A_i & A_i\\ A_i & A_i\end{bmatrix}=\frac{1}{2^{i+2}}A_{i+1}.\square$$

Por lo tanto, a partir de los valores del vector $\tilde{F}_k$, el vector de los coeficientes de la representación F_k, pueden ser calculados por una simple multiplicación matriz-vector como sigue:

$$F_k=\frac{1}{2^{k+1}}A_k\tilde{F}_k \tag{11}$$

3.2. Optimización simbólica

Usando la representación canónica, una función general g(x) definida sobre un dominio discreto finito puede ser

optimizada por el algoritmo presentado en la sección 2.2.

El vector de coeficientes de la representación polinomial $f_0(u_0,\ldots,u_{k-1},u_k) = f(u_0,\ldots,u_{k-1},u_k)$ puede ser calculado directamente por la fórmula (11), notando que:

$$\tilde{F}_k = \left[f(-1,\cdots,-1,-1),\ f(-1,\cdots,-1,1),\ \cdots, f(1,\cdots,1,1) \right]^T$$
$$= \left[g(0),\ g(1),\ \cdots,\ g(2^{k+1}) \right]^T$$

Entonces, u_k^+ puede ser calculado por:

$$u_k^+(u_0,\ldots,u_{k-1}) = -\text{Sign}(f_0(u_0,\ldots,u_{k-1},-1) - f_0(u_0,\ldots,u_{k-1},1))$$

La parte iterativa simbólica procede como sigue, para $i = k-1,\cdots,0$:

- Sustituir u_{i+1}^+ en $f_{k-i-1}(u_0,\cdots,u_{i+1})$ y obtenemos $f_{k-i}(u_0,\cdots,u_i)$.
- Evaluar $f_{k-i}(u_0,\cdots,u_i)$ sobre el dominio D_i (vector $\tilde{F}_i$).
- Generar la matriz A_i.
- Resolver el sistema de ecuaciones lineales:

$$\begin{bmatrix} -A_i & A_i \\ A_i & A_i \end{bmatrix} F_{i+1} = \tilde{F}_{i+1}.$$

- Generar la representación polinómica $f_{k-i}(u_0,\cdots,u_i)$ con coeficientes en el vector $\tilde{F}_i$
- Calcular

$$u_i^+(u_0,\ldots,u_{i-1}) = -\text{Sign}(f_{k-i}(u_0,\ldots,u_{i-1},1) - f_{k-i}(u_0,\ldots,u_{k-1},0)).$$

La segunda parte del algoritmo es explícita y procede hacia adelante para calcular la secuencia óptima $(u_0^*, u_1^*,\cdots,u_k^*)$ como en la parte 2 de la sección 2.2.

3.3. Mejoras en el algoritmo simbólico

El algoritmo que acaba de describirse en 3.2 es algo difícil de poner en ejecución debido a la necesidad de evaluar la función $f_{k-i}(u_0,\cdots,u_i)$ en todos los vértices del hipercubo $\{-1,1\}^{i+1}$ para obtener su representación polinómica canónica. Una simplificación importante sería alcanzada si es posible calcular directamente el vector del coeficiente, F_i, para la iteración i, a partir del vector de coeficiente, F_{i+1}, en la iteración i+1, sin tener que calcular el vector de valores $\tilde{F}_i$. Tal mejora se puede alcanzar usando el lema siguiente.

Definición: $x \in M_{n\times m}$ entonces $|x| = \left[|x|_1 \ \ |x|_2 \cdots |x|_n \right]^T$.

Lema 2.5:

La secuencia de los vectores de coeficientes $\{F_i\}$ de las funciones polinómicas canónicas f_{k-i} para i=k-1, …, 0 satisface la siguiente expresión de recurrencia:

$$F_i = \frac{-1}{2^{i+1}} A_i \left| A_i F_{i+1,1} \right| + F_{i+1,0}. \qquad (12)$$

Prueba

La i-esima fila de la matriz A_k puede ser expresada por $A_{k,i}$.

La función f_{k-i} de i+1 variables:

$$f_{k-i}(u_0,\ldots,u_i) = \sum_{\underline{\varepsilon}} f_{k-i;\varepsilon_0\cdots,\varepsilon_i} u_0^{\varepsilon_0}\cdots u_i^{\varepsilon_i},$$

con $\underline{\varepsilon} = (\varepsilon_0,\cdots,\varepsilon_i) \in \{0,1\}^{k+1}$, fue obtenido de:

$$f_{k-1-i}(u_0,\cdots,u_{i+1})$$

substituyendo en u_{i+1} por

$$u_{i+1}^+(u_0,\ldots,u_i) = -\text{Sign}\left[\sum_{\underline{\varepsilon}} f_{k-i-1;\varepsilon_0\cdots\varepsilon_i 1} u_0^{\varepsilon_0}\cdots u_i^{\varepsilon_i} \right].$$

Así, por otra parte:

$$f_{k-i} = -\left(\sum_{\underline{\varepsilon}} f_{k-i-1;\varepsilon_0\cdots\varepsilon_i 1} u_0^{\varepsilon_0}\cdots u_i^{\varepsilon_i} \right) \text{Sign}\left[\sum_{\underline{\varepsilon}} f_{k-i-1;\varepsilon_0\cdots\varepsilon_i 1} u_0^{\varepsilon_0}\cdots u_i^{\varepsilon_i} \right] + \sum_{\underline{\varepsilon}} f_{k-i-1;\varepsilon_0\cdots\varepsilon_i 0} u_0^{\varepsilon_0}\cdots u_i^{\varepsilon_i} \qquad (13)$$

El correspondiente vector de valores $\tilde{F}_i$ puede ser calculado de (13) por:

$$\tilde{F}_i = \begin{pmatrix} f_{k-i}(-1,\cdots,-1) \\ \vdots \\ f_{k-i}(1,\cdots,1) \end{pmatrix}$$

$$= \begin{pmatrix} A_{i,1}F_{i+1,1}\text{Sign}\left[A_{i,1}F_{i+1,1} \right] + A_{i,1}F_{i+1,0} \\ \vdots \\ A_{i,2^{i+1}}F_{i+1,1}\text{Sign}\left[A_{i,2^{i+1}}F_{i+1,1} \right] + A_{i,2^{i+1}}F_{i+1,0} \end{pmatrix} = A_i F_i$$

$$= -\begin{pmatrix} \left|A_{i,1}F_{i+1,1}\right| + A_{i,1}F_{i+1,0} \\ \vdots \\ \left|A_{i,2^{i+1}}F_{i+1,1}\right| + A_{i,2^{i+1}}F_{i+1,0} \end{pmatrix},$$

que es,

$$\tilde{F}_i = \left|A_i F_{i+1,1}\right| + A_i F_{i+1,0} = A_i F_i .$$

Por lo tanto, usando el Lema 2.4, la recursión del Lema 2.5 es obtenida multiplicando ambos términos de esta igualdad por la inversa de A_i.

Usando el Lema 2.5, la secuencia de vectores de coeficientes $\{F_i\}$ puede ser calculado desde el vector de coeficientes $\{F_k\}$ de las funciones polinómicas canónicas $f_0(u_0,\ldots,u_{k-1},u_k) = f(u_0,\ldots,u_{k-1},u_k)$.

Entonces, las expresiones formales (óptimas) de las variables $u_i^+(u_0,\ldots,u_{i-1})$ pueden ser calculadas ahora como se describe en la sección 3.2 con la ecuación(12). Así, al algoritmo anteriormente descrito lo denominamos SCDO, por sus siglas en ingles: Symbolic Computation in Discrete Optimization.

La complejidad de la manipulación simbólica de la aplicación polinómica sobre el espacio producto es bajo. La substitución en los monomios $u_0u_1\cdots u_k$, es de complejidad *O*(k). La substitución en $f(u_0,\ldots,u_{k-1},u_k)$ en la representación canónica es un producto escalar del vector de monomios por el vector de coeficientes, el cual significa *O*(n) "elementales" operaciones, donde $n = 2^{k+1}$. En este contexto "elementales" significa, por ejemplo operaciones simbólicas con monomios y polinomios de la forma $\sum C_{\underline{\varepsilon}} u^{\underline{\varepsilon}}$, donde $\underline{\varepsilon}$ es un multi-índice vectorial de 0,1 . Esa complejidad está también *O*(n), por lo tanto la complejidad completa de la manipulación simbólica es *O*(n). La asignación del Signo es realizada por un producto matriz-vector entre escalares que también tienen complejidad *O*(n). En relación con las complejidades mencionadas, el algoritmo para minimizar la función sobre $\{-1,1\}^{k+1}$, tiene complejidad *O*(n).

4 Principio de mínimo sobre dominios finitos

En el caso de sistemas sobre dominios finitos tal como en el caso de sistemas a eventos discretos el principio de mínimo genera como condición necesaria en cada etapa de la optimización, para el caso monocriterio una desigualdad variacional o un conjunto de desigualdades variacionales para el caso multicriterio con dos indeterminadas. Tal como lo muestra el siguiente teorema:

Teorema: Principio de Mínimo de Pontryaguin sobre Dominios Finitos (PMPDF).

Dado:

$$x(k+1) = f_k(x(k),u(k)),\ x(0)=x_0,$$

con $X_i \subset X \subset Z^n$ $U_i \subset U$ y una función de costo multiobjetiva, $J_1(x,u) = \Phi(x(K))$, con $\Phi : X_K \to Z^m$ definida sobre el conjunto los procesos discretos, bajo el supuesto que Z^m está equipada con una relación de orden dada por : $y_1 \le y_2$.

Suponga que (x^*,u^*) es un proceso óptimo de longitud K.

Si $u^*(0) = u(0), u^*(1) = u(1),\ldots,\ u^*(i-1) = u(i-1)$, $u(i) = v$, $u^*(i+1) = u(i+1)$, …, $u^*(K-1) = u(K-1)$, es el control y $x_{i,v}$ es la trayectoria correspondiente, $p_{i,v}$ es la solución de la ecuación adjunta correspondiente al par de procesos, (x^*,u^*), (x,u), entonces se cumple que :

$$H(i,x^*(i),u^*(i),p_{i,v}(i+1)) \le H(i,x^*(i),v,p_{i,v}(i+1)) \quad (14)$$

Podemos reescribir la ecuación en diferencia del sistema y la ecuación adjunta en términos del Hamiltoniano como :

$$x^*(i+1) = R_p H(i,x^*(i),u^*(i),p_{i,v}(i+1)),$$
$$p_{i,v}(i+1) = R_x H(i,x^*(i),u^*(i),p_{i,v}(i+1)) .$$

Así, el dualismo Hamiltoniano aquí encontrado es equivalente al del cálculo de variaciones.

Podemos reescribir la expresión (14) de dos indeterminadas como, para todo v existe u^* tal que:

$$p_{i,v}(i+1)\left[f_i(x^*(i),v) - f_i(x^*(i),u^*(i))\right] \ge 0 , \qquad (15)$$

se satisface. El u^* obtenido es la solución al problema de síntesis planteado. Como es bien sabido, un único método para obtener u^* en la desigualdad variacional (15) no existe. Como u^* y $v \in Z$ podemos, en principio, transformar este problema en un problema min-max sobre el dominio {-1,1}. Las solución se realiza en dos fases, en la primera fase un mínimo es calculado para v y en la segunda fase un máximo es calculado para u^*. En ambos cálculos el algoritmo SCDO detallado anteriormente es usado. Esta solución es detallada a continuación.

4.1 Algoritmo SCDO en procedimiento min-max

• Transformar la desigualdad sobre el dominio {-1,1}, una forma posible es usar la representación diádica siguiente:

$v=1/2((1+v_0)+2(1+v_1)+ ... +2^k(1+v_k))$ y $u^*=1/2((1+u^*_0)+2(1+u^*_1)+ ... +2^k(1+u^*_k))$.

• Aplicar el algoritmo SCDO sobre las variables v. Aquí el resultado es una función con indeterminada u* y parámetro x*.
• A la función obtenida en 2 se le debe calcular el máximo. Para ello se cambia el signo a la función y podemos aplicar el algoritmo SCDO de nuevo.
• Aplicar el algoritmo SCDO sobre las variables u*, y se obtiene una expresión paramétrica en x*.
• Invertir la transformación diádica, para obtener el resultado en coordenadas originales parametrizada x*, como $u^*(.) \in \Psi\left(.,x^*(.)\right)$.

Observaciones:

• Independientemente de la cardinalidad de u* y de v, solo el dominio original debe ser transformado en el dominio {-1,1} de tal manera que los resultados son expresados en una secuencia de 1 y –1, parametrizadas por x*.
• En principio k+1 valores mínimos tiene que ser obtenidos y que representan una expresión o valor en coordenadas originales de la solución de la desigualdad variacional.
• Más de un valor óptimo se obtiene, si el caso 2 ocurre como se describe en 2.2..

5 Programación dinámica parametrica sobre dominios finitos

Como el método de Programación Dinámica es un método de descomposición directa en donde se reemplaza la optimización de una función de varias variables por la resolución recursiva del problema de optimización de una sola variable entonces, la programación dinámica provee un método de optimización multietapa a través de la resolución de la ecuación funcional:

$$J_k(x_k,p) = \min_{u_k}\left\{c_k(x_k,u_k,p) + J_{k+1}(x_{k+1}.p)\right\} = \min_{u_k}\left\{c_k(x_k,u_k,p) + J_{k+1}(f_k(x_k,u_k,p),p)\right\}, \quad (16)$$

para k=K hasta 0, con la condición terminal:

$$J_K(x_K,p) = c_K(x_K,p). \quad (17)$$

Clásicamente, el control óptimo de la última etapa y la función de costo dada por (17) son evaluados primero y el algoritmo de programación dinámica es aplicado hacia atrás, usando la ecuación de optimalidad (16). Por construcción $J_{k+1}(x_{k+1},p)$, es el costo óptimo "para-ir-a" desde el estado x_{k+1} en periodo k+1.

En el enfoque paramétrico propuesto, la evaluación se realiza en cada etapa ya el método proporciona las expresiones formales del control óptimo actual, u_k^*, y por ende la del costo óptimo actual, $J_k(x_k,p)$, que dependen del estado actual y de los parámetros exógenos. Así, el método produce un control realimentado óptimo: $u_k(x_k,p)$, y el valor de la función objetivo $J_0(x_0,p)$.

Debido a la característica discreta del conjunto de los estados y de las decisiones, la resolución de la ecuación funcional (16) contiene una enumeración y una evaluación implícitas de todos los caminos asociados a posibles trayectorias. Sin embargo, no hay necesidad de enumerar todos los valores posibles de los parámetros. Ésta es la principal ventaja de usar expresiones simbólicas en forma cerrada del control óptimo y el valor óptimo de la función en cada etapa.

5.1 Algoritmo para la programación dinámica paramétrica (PDPA)

El algoritmo para el método de programación dinámica paramétrica (PDPA), basado en optimización simbólica, es como sigue:

Considere:

$$J_K^+(x_K,p) = c_K(x_K,p).$$

Para k=K-1 to 0, calcular:

$$J_k(x_k,u_k,p) = \operatorname{Min}_{u_k}\left[c_k(x_k,p) + J_{k+1}^+(f_k(x_k,u_k,p),p)\right] \quad (18)$$

Como:

$$g_k\left(x_k,u_k,p\right) = c_k\left(x_k,u_k,p\right) + J_{k+1}^+\left(f_k\left(x_k,u_k,p\right),p\right)$$

se aplica la transformación diádica (TD), lema 2.1, esto nos da:

$$g_k\left(x_k,u_{k1},\cdots,u_{kq},p\right) = c_k\left(x_k,u_{k1},\cdots,u_{kq},p\right) + J_{k+1}^+\left(f_k\left(x_k,u_{k1},\cdots,u_{kq},p\right),p\right).$$

Se aplica el algoritmo SCDO en el paso k, y se obtiene:

$$\left(u_{k1}^+\left(x_k,p\right),\cdots,u_{kq}^+\left(x_k,p\right)\right) \xrightarrow{TD^{-1}} u_k^+\left(x_k,p\right),$$

donde:

$$u_{k\,i}^+(x_k,p) = u_{k\,i}^+(x_k,u_{k\,1},\cdots,u_{k\,i-1},p).$$

Así, obtenemos:

$$J_k^+(x_k,u_k^+,p)=c_k(x_k,p)+J_{k+1}^+(f_k(x_k,u_k^+,p),p)=c_k(x_k,p)+I_k(x_k,p).$$

Si x_0^* es conocido entonces, podemos calcular k=0 hasta K-1:

$$u_k^*(x_k^*,p)=u_k^+(x_k^*,p),$$
$$J_k^*(x_k^*,u_k^*,p)=J_k^+(x_k^*,u_k^*,p),$$
$$x_{k+1}^*=f_k(x_k^*,u_k^*,p),$$

y finalmente obtenemos:

$$J_K^*\left(x_K^*,p\right)=c_K\left(x_K^*,p\right)$$
$$J_K^*(x_K^*,p)=c_K(x_K^*,p)$$

6 Ejemplo

El ejemplo que a continuación se muestra fue tratado por el PMPDF y por PDPA

Considere el siguiente sistema dinámico discreto parametrico:

$$x(k+1)=f_k\left(x(k),u(k),p\right),\quad x(0)=\xi,$$

con $p\in\Re^+$, y

$$X_0=\{-1,1\},X_1=\{-1,1\},X_2=\{-1,1\}\times\{-1,1\},U=\{-1,1\}$$

$$f_0(x,u,p)=\mathrm{Sign}\left((p+1)xu+px+3u+p^2-1\right),$$

$$f_1(x,u,p)=\begin{pmatrix}\mathrm{Sign}\left((0.5p^2+p-0.5)xu+(p^2+2p-1)x+\right.\\ \left.+(p^2+2p+2)u+2p^2+4p+4\right)\\ \mathrm{Sign}\left((0.5p^2)xu+(0.25p^2)x+\right.\\ \left.+(p^2+p+2)u+0.5p^2+0.5p+1\right)\end{pmatrix}$$

La función de costo es:

$$J=G\left(x_1(2),x_2(2),p\right)+g_1\left(x(1),u(1),p\right)+g_0\left(x(0),u(0),p\right)$$

El problema de control óptimo es definido por:

$$J^*=\min_{u(0),u(1)}\{g_0\left(x(0),u(0),p\right)+g_1\left(x(1),u(1),p\right)+$$
$$+G\left(x_1(2),x_2(2),p\right)\},$$

donde:

$$g_0(x,u,p)=(1+p)xu+0.5x+(p+0.5p^2)u+p,$$

$$g_1(x,u,p)=(p^2-0.5)xu+(p^2+p)x+(p^2+p+0.5)u+1,$$
$$G(x_1,x_2,p)=(1+p+2p^2)x_1+(p^2+1)x_2+p^2.$$

Se puede verifier que para $p\in\Re^+$:

$$\mathrm{Sign}\left[(0.5p^2+p-0.5)x(1)u(1)+(p^2+2p-1)x(1)+\right.$$
$$\left.+(p^2+2p+2)u(1)+2p^2+4p+4\right]=1$$

$$\mathrm{Sign}\left[(0.5p^2)x(1)u(1)+(0.25p^2)x(1)+\right.$$
$$\left.+(p^2+p+2)u(1)+.05p^2+0.5p+1\right]=u(1)$$

Si aplicamos PMPDF tenemos:

Las desigualdades variacionales nos quedan

Para i=1:

$$0\le\left[(p^2-0.5)x_1^*(1)+(2p^2+p+1.5)\right]\left(v(1)-u^*(1)\right)$$

La aplicación del algoritmo SCDO a la desigualdad nos da:

$$\mathrm{pol}\left(v(1),u^*(1)\right)=\left[(p^2-0.5)x_1^*(1)+(2p^2+p+1.5)\right]*\ldots$$
$$\left(v(1)-u^*(1)\right)$$

$$v^+(1)=-\mathrm{Sign}\left[(p^2-0.5)x_1^*(1)+(2p^2+p+1.5)\right]$$

Substituyendo la desigualdad nos queda:

$$-\mathrm{pol}\left(v^+(1),u^*(1)\right)=-\mathrm{pol}\left(u^*(1)\right)=$$
$$=\left[(p^2-0.5)x_1^*(1)+(2p^2+p+1.5)\right]$$
$$\left(\mathrm{Sign}\left[(p^2-0.5)x_1^*(1)+(2p^2+p+1.5)\right]+u^*(1)\right)$$

$$u^*(1)=-\mathrm{Sign}\left[(p^2-0.5)x_1^*(1)+(2p^2+p+1.5)\right]$$

Para i=0:

$$0\le\left[(p^2-0.5)u^*(1)+(p^2-p)\right]$$
$$\left(\mathrm{Sign}\left((p+1)x_1^*(0)v(0)+px_1^*(0)+3v(0)+p^2-1\right)-\right.$$
$$\left.\mathrm{Sign}\left((p+1)x_1^*(0)u^*(0)+px_1^*(0)+3u^*(0)+p^2-1\right)\right)+$$
$$\left[(1+p)x_1^*(0)+(p+0.5p^2)\right]\left(v(0)-u^*(0)\right)$$

La aplicación del algoritmo SCDO a la desigualdad nos da:

$$\text{pol}\left(v(0),u^*(0)\right)=\left[(p^2-0.5)u^*(1)+(p^2-p)\right]$$
$$\left(\text{Sign}\left((p+1)x_1^*(0)v(0)+px_1^*(0)+3v(0)+p^2-1\right)-\right.$$
$$\left.-\text{Sign}\left((p+1)x_1^*(0)u^*(0)+px_1^*(0)+3u^*(0)+p^2-1\right)\right)+$$
$$\left[(1+p)x_1^*(0)+(p+0.5p^2)\right]\left(v(0)-u^*(0)\right)$$
$$v^+(0)=\text{Sign}\left[\left[(p^2-0.5)u^*(1)+(p^2-p)\right]\right.$$
$$\left(\text{Sign}\left(-x_1^*(0)+p^2-4\right)-\text{Sign}\left((2p+1)x_1^*(0)+p^2+2\right)-\right.$$
$$\left.\left.-2\left[(1+p)x_1^*(0)+(p+0.5p^2)\right]\right]\right.$$
$$u^*(0)=-\text{Sign}\left[\left[(p^2-0.5)u^*(1)+(p^2-p)\right]\right.$$
$$\left(\text{Sign}\left((2p+1)x_1^*(0)+p^2+2\right)-\text{Sign}\left(-x_1^*(0)+p^2-4\right)\right)+$$
$$\left.+2\left[(1+p)x_1^*(0)+\right.\right.$$

Si aplicamos PDPA tenemos**:**

$$J_1(x(1),p)=\min_{u(1)}\left\{g_1(x(1),u(1),p)+G(x(2),p)\right\},$$
$$J_0(x(0),p)=\min_{u(0)}\left\{g_0(x(0),u(0),p)+J_1(x(1),p)\right\}.$$

Por substitución:

$$J_1(x(1),p)=\min_{u(1)}\left\{(p^2-0.5)x(1)\,u(1)+(p^2+p)x(1)+\right.$$
$$\left.+(2p^2+p+1.5)u(1)+(2+p+3p^2)\right\}.$$

Así, el control óptimo paramétrico es:

$$u_1^+(x(1),p)=-\text{Sign}\left[(p^2-0.5)x(1)+(2p^2+p+1.5)\right].$$

El valor óptimo de J_1 es:

$$J_1^+(x(1),p)=\left\{-\left|(p^2-0.5)x(1)+(2p^2+p+1.5)\right|+(p^2+p)x(1)+(2+p+3p^2)\right\}.$$

Por substitución, el valor óptimo de J_0 es:

$$J_0^+(x(0),p)=\min_u\left\{(1+p)x(0)\,u+0.5x(0)+(p+0.5p^2)u+p+\right.$$
$$\left\{-\left|(p^2-0.5)\text{Sign}\left((p+1)x(0)u+\right.\right.\right.$$
$$\left.\left.\left.px(0)+3u+p^2-1\right)+2p^2+p+1.5\right|\right\}+(p^2+p)$$
$$\text{Sign}\left[(p+1)x(0)u+px(0)+3u+p^2-1\right]$$
$$\left.+3p^2+p+2\right\},$$

Así, el control óptimo paramétrico es:

$$u_0^+(x(0),p)=\text{Sign}\left[-2\left((1+p)x(0)\right)-2(p+0.5p^2)+(p^2+p)\right.$$
$$\left\{\text{Sign}\left[-x(0)+p^2-4\right]-\text{Sign}\left[(2p+1)x+p^2+2\right]\right\}+$$
$$\left.+\left|(p^2-0.5)\text{Sign}\left[(2p+1)x(0)+p^2+2\right]+2p^2+p+1.5\right|-\right.$$
$$\left.-\left|(p^2-0.5)\text{Sign}\left[-x(0)+p^2-4\right]+2p^2+p+1.5\right|\right].$$

Entonces, para $x^*(0)$ dado:

$$u^*(0)=u_0^+(x^*(0),p),$$
$$x^*(1)=\text{Sign}\left[(p+1)x(0)u^*(0)+px(0)+3u^*(0)+p^2-1\right]$$

$$J_1^*(x^*(1),p)=J_1^+(x^*(1),p),$$

$$u_1^*(x^*(1),p)=-\text{Sign}\left[(p^2-0.5)x^*(1)+(2p^2+p+1.5)\right]$$

$$x^*(2)=\begin{pmatrix}\text{Sign}\left((0.5p^2+p-0.5)x^*(1)u^*(1)+\right.\\ \quad+(p^2+2p-1)x^*(1)\\ \quad\left.+(p^2+2p+2)u^*(1)+2p^2+4p+4\right)\\ \text{Sign}\left((0.5p^2)x^*(1)u^*(1)+(0.25p^2)x^*(1)+\right.\\ \quad\left.+(p^2+p+2)u^*(1)+0.5p^2+0.5p+1\right)\end{pmatrix}$$

$$J_2^*(x^*(2),p)=G(x^*(1),x^*(2),p).$$

Para ambos procedimientos el control óptimo y la trayectoria óptima para p=0 y p=5 es como sigue:

p=0:

- Si $x^*(1)=\pm1$, entonces $u^*(1)_0=-1$.
- Si $x_1^*(0)=-1$, entonces $u^*(0)_0=1$
- si $x_1^*(0)=1$, entonces $u^*(0)_0=-1$.

p=5:

- Si $x^*(1)=\pm1$, entonces $u^*(1)_5=-1$.
- Si $x^*(1)=\pm1$, entonces.

7 Conclusiones

- El algoritmo de optimización simbólico propuesto es aplicable para funciones generales definidas sobre un dominio discreto finito.
- Aquí, el mínimo absoluto de funciones Booleanas es dado por una fórmula explícita. El uso de esta fórmula es pro-

puesta como una alternativa a los métodos de búsqueda clásica.

- El método está compuesto por dos partes. La primera es un cálculo recursivo hacia atrás , en donde se obtienen las expresiones implícitas de las variables Booleanas.. La segunda parte, instancia progresivamente las expresiones explicitas de los valores optimales de las variables
- La complejidad del algoritmo es comparable con los métodos numéricos de tipo enumerativo. Pero el algoritmo propuesto tiene la ventaja de poderse aplicar directamente a los problemas en los cuales tanto las decisiones como la función de costo asociadas dependen de parámetros exógenos.
- La complejidad de la manipulación simbólica de la aplicación polinómica sobre el espacio producto es bajo. La substitución en los monomios $u_0 u_1 \cdots u_k$, es de complejidad $O(k)$. La substitución en $f(u_0,\ldots,u_{k-1},u_k)$ en la representación canónica es un producto escalar del vector de monomios por el vector de coeficientes, el cual significa $O(n)$ "elementales" operaciones, donde $n = 2^{k+1}$. En este contexto "elementales" significa, por ejemplo operaciones simbólicas con monomios y polinomios de la forma $\sum C_{\underline{\varepsilon}} u^{\underline{\varepsilon}}$, donde $\underline{\varepsilon}$ es un multi-índice vectororial de $0,1$. Esa complejidad es también $O(n)$, por lo tanto la complejidad completa de la manipulación simbólica es $O(n)$. La asignación del Signo es realizada por un producto matriz-vector entre escalares que también tienen complejidad $O(n)$. En relación con las complejidades mencionadas, el algoritmo para minimizar la función sobre $\{-1,1\}^{k+1}$, tiene complejidad $O(n)$.
- Para obtener la solución del conjunto de desigualdades variacionales, para el PMPDF, es necesario aplicar un algoritmo basado en el cálculo simbólico. Este puede ser considerado como una generalización de la ecuación de Ricatti.
- Como nuestro método es una extensión de las técnicas analíticas aplicadas a sistemas continuos, ella puede ser aplicada a los sistemas continuos e híbridos sin ninguna hipótesis adicional de convexidad sobre la parte discreta del modelo.
- La combinación de la programación dinámica con cálculo formal es un enfoque prometedor en la caso de sistemas dinámicos conducidos por parámetros exógenos de evolución. El hecho de que pocos autores han explorado este enfoque es probablemente debido a la complejidad combinatoria de la programación dinámica, junto con la dificultad para proporcionar las expresiones explícitas para el control óptimo y para la función valuada. Sin embargo, estas limitaciones se pueden superar para algunas clases de los sistemas a discretos y sistema a eventos discretos. Adicionalmente presenta una sustancial ventaja sobre el procedimiento clásico.

Referencias

Boulehmi M, 1999, Mise en oeuvre et évaluation d'algorithmes d'optimisation et de commande optimale sur un système de cacul formel, Doctorate thesis.

Bellman R, 1957, Dynamic programming, Princeston University Press, N.Y.

Calderón J, Chacón E y Becerra L, 1999, Gas process automation in petroleum production, Proc. 3rd. International Conference on Industrial Automation, Montreal Canada, pp. 2.5-2.7.

Cardillo J, Szigeti F, Hennet J-L y Calvet J-C, 2001, Symbolic computation in discrete optimization, LAAS report, submitted to Journal of Symbolic Computation.

Gimenez JL, 1989, Contribition a la decomposition de systemes interconectes par programmation dynamique non seriellle, Application a des systemes de puissance, Doctorate thesis, Toulouse, France.

Cardillo-Albarrán JJ, 1994, Optimización de sistemas a Eventos discretos usando el principio de mínimo de Pontryagin, Thesis Msd, Fac. Engineering, ULA, Mérida.

Cardillo-Albarrán JJ y Szigeti F, 1995, A symbolic algorithm for the optimization of integer processes over finite state space, International Congress on Industrial and Applied Mathematics, ICIAM 95, Hamburg, Germany, 3-7, July.

Larson R y Casti J, 1978, Principle of dynamic programming, Control and System Theory, Part I and II, Marcel Dekker, ISBN 0-8247-6589-3.