Benemérita Universidad Autónoma de Puebla

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS
LICENCIATURA EN MATEMÁTICAS APLICADAS

TESIS
PARA OBTENER EL GRADO DE
LICENCIADO EN MATEMÁTICAS APLICADAS

Un acercamiento a problemas de paro óptimo vía programación dinámica

Erika Patricia Domínguez Rios

Asesor Hugo Adán Cruz Suárez

2020

A mis padres.

Introducción

Los procesos de decisión o control Markovianos modelan sistemas dinámicos estocásticos, es decir, sistemas cuya evolución esta sujeta a factores aleatorios y que puede modificarse por medio de la selección de ciertas variables de decisión o de control, dichos procesos son muy útiles en áreas como economía donde son usados para optimizar problemas de crecimiento económico [6]. El problema de control óptimo consiste en encontrar una política que optimice el criterio de rendimiento, el cual, en este caso será el criterio de costo total esperado, una política es una regla mediante la cual se elige una acción en cada punto de observación del proceso. Una manera de resolver un proceso de decisión de Markov es basándose en el principio de optimalidad de Bellman conocido como programación dinámica.

Bellman [5] usa el nombre "proceso de decisión Markoviano" para referirse a un programa dinámico cuyo mecanismo de evolución es aleatorio y satisface una propiedad markoviana o de pérdida de memoria. El propósito del presente trabajo es dar una introducción breve y elemental a esta clase de procesos considerando modelos con espacio de estados finitos.

En el primer capítulo se presentan conceptos básicos de procesos estocásticos y un pequeño problema de inventario, en el segundo capítulo se da una breve introducción a la programación dinámica y con ello se prueba el teorema principal de la tesis, conocido como el teorema de la programación dinámica, para posteriormente extender el problema de inventario del primer capitulo y ejemplificar el teorema probado. En el tercer y último capítulo se exponen problemas de paro óptimo con una introducción y el desarollo de dos de estos, asimismo se presentan los pseudocódigos de cada uno de ellos y simulaciones realizadas en Python. En los anexos se encuentra el código fuente de cada problema del que se realizó una simulación.

Índice general

1 Programación Dinámica

En 1953 Richard Bellman propone la programación dinámica como un método de optimización para resolver problemas en procesos de decisión en múltiples pasos, sin embargo, actualmente es más amplia la cantidad de usos que esta puede tener en problemas tanto prácticos como teóricos. Este capítulo es la médula del trabajo de tesis, en el se presentan dos secciones. La primera sección (Sección 1.1) contiene definiciones y notaciones de uso común y recurrentes a lo largo de la tesis, en la segunda sección se encuentra un ejemplo, el cual será referido en el siguiente capítulo.

1.1 Preliminares

Para la comprensión del trabajo es necesario conocer conceptos básicos de procesos estocásticos, en este capítulo se presentan conceptos o nociones que serán de utilidad, sin suponer que han sido conocidos anteriormente. y posteriormente conceptos fundamentales de programación dinámica.

Definición 1.1. *(i) Un proceso estocástico es una sucesión de variables aleatorias* $\{X_t : t \in T\}$ *definidas en un espacio de probabbilidad* $(\Omega, \mathcal{F}, \mathbb{P})$*, donde* T *es un conjunto de índices y* X *el rango común de las variables aleatorias (v.a.) a los cuales se les denomina espacio tiempo y espacio de estados, respectivamente.*

(ii) Un proceso de Markov se define como un proceso estocástico que cumple la siguiente propiedad: para $x_0, x_1, \ldots, x_{n-1}, x, y \in X$ *y* $n \geq 1$*,*

$$\mathbb{P}\left(X_{n+1} = y | X_0 = x_0, X_1 = x_1, \ldots, X_n = x\right) = \mathbb{P}\left(X_{n+1} = y | X_n = x\right).$$

Sea $\{X_t : t = 0, 1, 2, \ldots\}$ un proceso de Markov homogéneo definido en un espacio de probabilidad $(\Omega, \mathcal{F}, \mathbb{P})$ y con espacio de estados $X \subset \mathbb{Z}$. Se supone que X_0 tiene una distribución conocida π_0, es decir,

$$\pi_0(y) = \mathbb{P}(X_0 = y),\ y \in X,$$

donde π_0 se denomina distribución inicial.

Notación 1.2. *En muchas aplicaciones de las cadenas de Markov el estado inicial es conocido, por ejemplo, suponga que $X_0 = x \in X$, en este caso $\pi_0(x) = 1$, y se dirá que la distribución de X_0 se concentra en el estado x.*

Notación 1.3. *Sean $x, y \in X$, se denota la transición en un paso de x a y del proceso estocástico por:*

$$P_{xy} = \mathbb{P}(X_1 = y | X_0 = x).$$

Definición 1.4. *Sea $c : X \longrightarrow \mathbb{R}^+$ la función de costo en un paso y $\{X_t\}$ un proceso de Markov.*

(i) La función de valor que caracteriza al costo total esperado hasta un horizonte $T \in \mathbb{Z}^+$, se define como:

$$V_T(x) := \mathbb{E}_x \left[\sum_{t=0}^{T-1} c(X_t) \right], x \in X.$$

(ii) Para $0 \leq t \leq T-1$, la función de valor truncado que caracteriza al costo total esperado hasta la etapa t es:

$$V_t(x) = \mathbb{E}_x \left[\sum_{n=0}^{t} c(X_n) \right], \ x \in X. \tag{1.1}$$

Notación 1.5. *Para $t = 0$ en (ii) de la Definición 1.4, para todo $x \in X$*

$$\begin{aligned} V_0(x) &= \mathbb{E}[c(X_0) | X_0 = x] \\ &= c(x). \end{aligned}$$

Para la demostración del próximo teorema será necesaria tener presente la siguiente identidad:

$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]] = \sum_y \mathbb{E}[X|Y = y]\mathbb{P}(Y = y). \tag{1.2}$$

Teorema 1.6. *La función de valor truncado V_t en (1.1) con $1 \leq t \leq T-1$ satisface:*

$$V_t(x) = c(x) + \mathbb{E}_x[V_{t-1}(X_1)]. \tag{1.3}$$

Demostración. Sea $x \in X$ y $t \in \{1, 2, \ldots, T-1\}$, fijos. Usando (ii) de la Definición 1.4 y propiedades de esperanza condicional (1.2),

$$\begin{aligned} V_t(x) &= c(x) + \mathbb{E}_x\left[\sum_{n=1}^{t} c(X_n)\right] \\ &= c(x) + \mathbb{E}\left[\mathbb{E}_x\left[\sum_{n=1}^{t} c(X_n)\right]\Big| X_1\right] \\ &= c(x) + \sum_{y\in S} \mathbb{E}\left[\sum_{n=1}^{t} c(X_n)\Big| X_0 = x, X_1 = y\right] \mathbb{P}_x(X_1 = y) \\ &= c(x) + \sum_{y\in S} \mathbb{E}\left[\sum_{n=1}^{t} c(X_n)\Big| X_0 = x, X_1 = y\right] \mathbb{P}_x(X_1 = y) \end{aligned}$$

Usando la propiedad de Markov y un cambio de varible $k = n - 1$ se siguen las identidades,

$$\begin{aligned} V_t(x) &= c(x) + \sum_{y\in S} \mathbb{E}_y\left[\sum_{k=0}^{t-1} c(X_k)\right] \mathbb{P}_x(X_1 = y) \\ &= c(x) + \sum_{y\in S} \mathbb{E}_y\left[V_{t-1}(X_1)\right] \mathbb{P}_x(X_1 = y) \end{aligned} \tag{1.4}$$

Por lo tanto, usando la definición de esperanza y como x y t son arbitrarios el resultado se sigue. ■

1.2 Ejemplo de inventario

Para reafirmar lo establecido tanto en el Teorema 1.6 como en las definiciones que le preceden, a continuación se muestra un ejemplo de un problema de inventario dónde se desea conocer el costo total esperado hasta cierta etapa establecida. Para facilitar el entendimiento de dicho ejemplo se presentan resultados obtenidos en Python.

Ejemplo 1.7. *Consideremos X_t el stock de un inventario al tiempo t, el cual sigue la dinámica:*

$$X_{t+1} = (X_t - \xi_{t+1})^+, t = 0, 1, \ldots \tag{1.5}$$

donde ξ_t representa la demanda en el periodo t, suponemos que la sucesión $\{\xi_t\}$ esta conformada por variables aleatorias independientes e idénticamente distribuidas (iid). Se considera que el almacén tiene una capacidad finita M, de esta manera, el espacio de estados es $X = \{0, 1, 2, \ldots, M\}$, si la demanda excede la cantidad de producto existente, se paga una penalización y no hay pedidos pendientes.
Se establece:

- $h \equiv$ *Costo de almacenaje por unidad.*

- $p \equiv$ *Costo de penalización (deuda de producto).*

En este caso consideraremos la siguiente función de costo, sin olvidar que para todo $a,\ b \in \mathbb{R},\ \max\{a,\ b\} = \frac{|a-b|+(a+b)}{2}$*, de esta manera,*

$$\begin{aligned} c(x) &= \mathbb{E}[h\,\max\{0, x-\xi\} + p\,\max\{0, -x+\xi\}] \\ &= h \cdot \mathbb{E}_x\left[\frac{|\xi - x| + x - \xi}{2}\right] + p \cdot \mathbb{E}_x\left[\frac{|x-\xi| + \xi - x}{2}\right] \\ &= \frac{h}{2} \cdot \Big[\mathbb{E}_x[|\xi - x|] + x - \mathbb{E}_x[\xi]\Big] + \frac{p}{2} \cdot \Big[\mathbb{E}_x[|\xi - x|] + \mathbb{E}_x[\xi] - x\Big] \\ &= \left(\frac{h-p}{2}\right)(x - \mathbb{E}[\xi]) + \left(\frac{h+p}{2}\right)\mathbb{E}[|\xi - x|], \end{aligned} \tag{1.6}$$

donde el operador esperanza, $\mathbb{E}$*, es con respecto a la variable aleatoria* ξ*.*
Se requiere el costo descontado total esperado de la Definicion 1.4 para 6 etapas, bajo las condiciones: $M = 5$ *,* $h = 3$*,* $p = 5$*, por lo que* $X = \{0, 1, 2, 3, 4, 5\}$ *y la distribución de la demanda está dada en la Tabla 1.1.*

d	0	1	2	3	4	5
P_d	0	0.1	0.1	0.3	0.3	0.2

Tabla 1.1: Distribución de la demanda

Haciendo uso del Teorema 1.6 para 6 *etapas, se presenta el Pseudocódigo 1 para este problema con espacio de estados* $X = \{0, 1, \ldots, 5\}$ *y con la distribución de* ξ *de la Tabla 1.1, para esta distribución, el valor esperado es* 3.4*. Para hacer uso de la ecuación (1.6) es necesario considerar la esperanza* $\mathbb{E}[|\xi - x|]$*, la cual estará representada por* $value2$*, en el cual, cada una de sus entradas significa el valor de* x *en la diferencia, es decir, en la entrada* 2*, se tiene* $\mathbb{E}[|\xi - 2|]$ *y* $\mathbb{E}[\xi] = value$*. Debido a la Notación 1.5 se tiene* $V_0(x) = c(x)$*, para toda* $x \in X$*, por lo que se ejecutan las lineas* 22 *y* 23*.*

Pseudocódigo 1 Stock de un inventario (a)

```
1:  procedure FUNTION(XI,X,x,bef)
2:      sum=0
3:      for y in X do
4:          a=x-y
5:          if 0 ≤ a ≤ 5 then
6:              sum =sum+ bef[y]*XI[a]
7:          return (sum)
8:
9:  procedure COST(XI,x,value,value2)
10:     h=3
11:     p=5
12:     C1=(h-p)/2·(x-value)
13:     C2=(h+p)/2·(x-value2[x])
14:     return(C1+C2)
15:
16: N=6
17: X=[0,1,...,M]
18: XI=[0,0.1,0.1,0.3,0.3,0.2]
19: value=3.4
20: value2=[3.4,2.4,1.6,1.0,1.0,1.6]
21:
22: for x in X do
23:     V[x]=COST(XI,x,value,value2)
24:
25: for t in (1,...,N) do
26:     bef=copyvector(V)
27:     for x in X do
28:         V[x]=COST(XI,x,value,value2)+FUNTION(XI,X,x,bef)
29:     write("Num. etapa: ", t)
30:     print("V: ", V)
```

Implementando el pseudocódigo en Python, se obtiene la siguiente Tabla.

x	$V_0(x)$	$V_1(x)$	$V_2(x)$	$V_3(x$	$V_4(x)$	$V_5(x)$
0	17	17	17	17	17	17
1	12	13.7	13.7	13.7	13.7	13.7
2	7.8	10.7	10.87	10.87	10.87	10.87
3	4.4	11.4	11.94	11.957	11.957	11.957
4	3.4	13.32	14.828	14.891	14.8927	14.8927
5	4.8	14.92	18	18.2478	18.2558	18.25597

Tabla 1.2

Notación 1.8. *Obsérvese que el estado* 0 *corresponde a un estado absorbente, es decir, si para algún* $k \geq 1, X_k = 0$, $\mathbb{P}_x - casi\ seguramente\ (c.s)$ *entonces,* $X_n = 0$, $\mathbb{P}_x - c.s$, $\forall n \geq k$. *Este tipo de comportamiento no ocurre en la práctica, en este caso al modelo (1.5) puede ser modificado del modo siguiente:*

$$X_{t+1} = (X_t + A_t - \xi_{t+1})^+, \tag{1.7}$$

donde A_t *es una variable de control la cual representa los productos solicitados en el stock al tiempo* t. *De este modo (1.7) se denomina proceso de control de Markov para el modelo de inventarios propuestos. En lo subsecuente se presenta la teoría general.*

2 Teorema de la Programación Dinámica (PD)

La programación dinámica es una técnica simple, pero poderosa, que ha demostrado ser muy útil para la resolución de problemas de decisión multietápicos. La idea fundamental que subyace en el método de la programación dinámica es el principio del invariante encajado, según el cual un problema difícil de resolver, esta encajado en una clase de problemas más simples de resolver, lo que permite encontrar una solución al problema original, en la segunda sección (Sección 2.2) está enfocada en enunciar y demostrar el Teorema de la Programación Dinámica 2.5, el cual provee un algoritmo para encontrar una función de valor y una política óptima.

2.1 Introducción

Definición 2.1. *Un modelo de control de Markov se define de acuerdo a las siguientes componentes:*

$$(X, A, \{A(x) : x \in X\}, F, c) \tag{2.1}$$

donde, X es un conjunto numerable denominado el espacio de estados; A es un conjunto numerable denominado el conjunto de acciones o controles; $A(x) \subset A$ el conjunto de acciones admisibles para cada estado $x \in X$; $F : \mathbb{K} \times H \longrightarrow X$ la dinámica del sistema esta dada por

$$X_{t+1} = F(X_t, A_t, \xi_t), X_0 = x, t = 0, 1, 2, \ldots \tag{2.2}$$

donde $\mathbb{K} = \{(x, a) : x \in X, a \in A(x)\}$ y $H = Rango(\xi_t)$; $c : \mathbb{K} \longrightarrow \mathbb{R}^+$ representa la función de costo, la cual se supone conocida.

Para introducir el concepto de política, considérese un modelo de control de Markov y defina $\mathbb{H}_t$, el espacio de las historias observadas del proceso de control hasta el tiempo t, como

$$\begin{aligned} \mathbb{H}_0 &= X, \\ \mathbb{H}_t &= \mathbb{K} \times \mathbb{H}_{t-1}, \end{aligned}$$

para $t = 1, 2, \ldots, T$. Un elemento h_t de $\mathbb{H}_t$ llamado $t-$historia es un vector de la forma

$$(x_0, a_0, x_1, a_1, \ldots, x_{t-1}, a_{t-1}, x_t),$$

donde $(x_i, a_i) \in \mathbb{K}$ para $i = 0, \ldots, t-1$ y $x_t \in X$. Obsérvese que, para cada t, $\mathbb{H}_t$ es un subespacio de $\mathbf{H}_t := (X \times A)^t \times X$ y $\mathbf{H}_0 := X$.

Definición 2.2. *Una política aleatorizada o simplemente política es una sucesión $\pi = \{\pi_t, t = 0, 1, 2, \ldots, T-1\}$ de probabilidades de transición definidas sobre A, dada la historia del proceso $\mathbb{H}_t$ y satisface que: $\pi_t(A(x_t|h_t)) = 1$ para toda $h_t \in \mathbb{H}_t$ y $t = 0, 1, 2, \ldots, T$.*

Definición 2.3. *Se definen Π como el conjunto de posibles políticas y como criterio de rendimiento el costo total acumulado con horizonte finito con la política $\pi \in \Pi$, como:*

$$J(\pi, x) := \mathbb{E}_x^{\pi} \left[\sum_{t=0}^{N-1} c(X_t, A_t) + c_N(X_N) \right], \; x \in X. \tag{2.3}$$

Donde $\mathbb{E}_x^{\pi}$ representa el valor esperado cuando se utiliza la política π dado el estado inicial $x_0 = x$ y $c_N : X \longrightarrow \mathbb{R}$ la función de costo en la etapa final.

El problema del control de interés es minimizar el criterio de rendimiento de horizonte finito.

Notación 2.4. *Se denota a J^* la función de valores, es decir,*

$$J^*(x) := inf\{J(\pi, x) : \; \pi \in \Pi\}, \; \forall x \in X. \tag{2.4}$$

El problema de control es encontrar una política $\pi^* \in \Pi$ tal que

$$J(\pi^*, x) = J^*(x), \; \forall x \in X. \tag{2.5}$$

Para la siguiente sección será necesario tener presente las siguientes identidades:

$$\mathbb{E}\left[\mathbb{E}[X|Y_1]|Y_2\right] = \mathbb{E}\left[\mathbb{E}[X|Y_2]|Y_1\right] = \mathbb{E}[X|Y_1]. \tag{2.6}$$

2.2 Teorema de PD

Teorema 2.5. *Sean $J_0, J_1, \ldots, J_N$ funciones en X definidas (hacia atrás, desde t=N hasta t=0) por*

$$J_N(x) := c_N(x) \tag{2.7}$$

y para $t = N-1, N-2, \ldots, 0$.

$$J_t(x) := \min_{A(x)} \left[c(x,a) + \sum_{y \in X} J_{t+1}(y) \mathbb{P}(X_{t+1} = y | X_t = x, A_t = a) \right], \tag{2.8}$$

A_t es la variable de control, para minimizar suponemos que para cada $t = 0, \ldots, N-1$, existen selectores $f_t \in \mathbb{F}$ t.q. $f_t(x) \in A(x)$ alcanza el mínimo en (2.8) $\forall x \in X$ y $t = 0, \ldots, N-1$,

$$J_t(x) = c(x, f_t(x)) + \sum_{y \in X} J_{t+1}(y) \mathbb{P}(X_{t+1} = y | X_t = x, A_t = f_t(x)). \tag{2.9}$$

Entonces, la política (determinista markoviana) $\pi^* = \{f_0, \ldots, f_{N-1}\}$ es la óptima, y la función de valor J^* es igual a J_0, i.e.

$$J^*(x) = J_0(x) = J(\pi^*, x), \ \forall x \in X. \tag{2.10}$$

Demostración. Sea $\pi = \{\pi_t\}$ una política arbitraria y $C_t(\pi, x)$ corresponde al costo total esperado del tiempo t al tiempo N, dado el estado $X_t = x$ en el tiempo t, i.e. si $t = 0, 1, \ldots, N-1$,

$$C_t(\pi, x) := \mathbb{E}^\pi \left[\sum_{n=t}^{N-1} c(X_n, a_n) + c_N(X_N) \middle| X_t = x \right] \tag{2.11}$$

$$C_N(\pi, x) := \mathbb{E}^\pi \left[c_N(X_N) | X_N = x \right] = c_N(x). \tag{2.12}$$

Si se usa la política π y $X_t = x$, nótese que por (2.3) y (2.11)

$$J(\pi, x) = C_0(\pi, x). \tag{2.13}$$

Afirmación 2.6. *Para todo $x \in X$ y $t = 0, \ldots, N$,*

$$C_t(\pi, x) \geq J_t(x). \tag{2.14}$$

Donde la igualdad se cumple si $\pi = \pi^*$, es decir,

$$C_t(\pi^*, x) = J_t(x). \tag{2.15}$$

La prueba de la Afirmación 2.6 se realizará por inducción hacia atrás. Con $t = N$ en (2.12) y (2.7), resulta $C_N(\pi, x) = J_N(x) = c_N(x)$.
Ahora como hipótesis de inducción, se asume que para algún $t = N-1, \ldots, 0$,

$$C_{t+1}(\pi, x) \geq J_{t+1}(x),\ x \in X. \tag{2.16}$$

De esta manera,

$$\begin{aligned} C_t(\pi, x) &= \mathbb{E}^\pi \left[\sum_{n=t}^{N-1} c(X_n, A_n) + c_N(X_N) \middle| X_t = x) \right] \\ &= \mathbb{E}^\pi \left[c(X_t, A_t) + \sum_{n=t+1}^{N-1} c(X_n, A_n) + c_N(X_N) \middle| X_t = x) \right] \\ &= \mathbb{E}^\pi \left[\mathbb{E}^\pi \left[c(X_t, A_t) + \sum_{n=t+1}^{N-1} c(X_n, A_n) + c_N(X_N) \middle| X_{t+1} = y, A_t = a) \right] \middle| X_t = x \right]. \end{aligned}$$

Sea $\beta = \sum_{n=t+1}^{N-1} c(X_n, A_n) + c_N(X_N)$ y resolviendo el valor esperado externo,

$$\begin{aligned} C_t(\pi, x) &= \sum_{a \in A} \left[\mathbb{E}^\pi \left[c(X_t, A_t) + \beta \middle| X_{t+1} = y, A_t = a \right] \middle| X_t = x \right] \cdot \pi_{\mathbf{t}}(\mathbf{A_t} = \mathbf{a} | \mathbf{X_t} = \mathbf{x}) \\ &= \sum_{a \in A} \left[c(x, a) + \mathbb{E}^\pi \left[\beta \middle| X_{t+1} = y, A_t = a \right] \middle| X_t = x \right] \cdot \pi_t(A_t = a | X_t = x) \\ &= \sum_{a \in A} \left[c(x, a) + \mathbb{E}^\pi \left[\mathbb{E}^\pi \left[\beta \middle| X_{t+1} = y \right] \middle| X_t = x, A_t = a \right] \right] \cdot \pi_t(A_t = a | X_t = x). \end{aligned} \tag{2.17}$$

Notar que $\mathbb{E}^\pi \left[\beta \middle| X_{t+1} = y \right] = C_{t+1}(\pi, y)$, por lo que calculando el valor esperado de (2.17), se obtiene

$$C_t(\pi, x) = \sum_{a \in A} \left[c(x, a) + \sum_{y \in X} C_{t+1}(\pi, y) \mathbb{P}(X_{t+1} = y | X_t = x, A_t = a) \right] \cdot \pi_t(A_t = a | X_t = x). \tag{2.18}$$

Teniendo en cuenta la hipótesis de inducción (2.16) y (2.18) se sigue que

$$C_t(\pi, x) \geq \sum_{a \in A} \left[c(x,a) + \sum_{y \in X} J_{t+1}(y) \mathbb{P}(X_{t+1} = y | X_t = x, A_t = a) \right] \cdot \pi_t(A_t = a | X_t = x). \tag{2.19}$$

Con $\alpha = c(x,a) + \sum_{y \in X} J_{t+1}(y) \mathbb{P}(X_{t+1} = y | X_t = x, A_t = a)$, se obtienen las siguientes desigualdades

$$\begin{aligned} C_t(\pi, x) &\geq \sum_{a \in A} [\min_{A(x)} (\alpha)] \cdot \pi_t(A_t = a | X_t = x) \\ &= [\min_{A(x)} (\alpha)] \cdot \sum_{a \in A} \pi_t(A_t = a | X_t = x) \end{aligned} \tag{2.20}$$

Notar que $\sum_{a \in A} \pi_t \left(A_t = a | X_t = x \right) = 1$, lo cual conduce a

$$\begin{aligned} C_t(\pi, x) &\geq \min_{A(x)} \left[c(x,a) + \sum_{y \in X} J_{t+1}(y) \mathbb{P}(X_{t+1} = y | X_t = x, A_t = a) \right] \\ &= J_t(x). \end{aligned}$$

Por lo que queda demostrado (2.14). Para probar (2.15) se realiza el mismo procedimiento hasta (2.19) pues en la hipótesis de inducción habría de suponerse la igualdad, con lo cual

$$C_t(\pi^*, x) = \sum_{a \in A} \left[c(x,a) + \sum_{y \in X} J_{t+1}(y) \mathbb{P}(X_{t+1} = y | X_t = x, A_t = a) \right] \cdot \pi_t(A_t = a | X_t = x) \tag{2.21}$$

En (2.20) se da la igualdad pues la política es π^*, utilizando el mismo α se siguen las siguientes igualdades

$$\begin{aligned} C_t(\pi^*, x) &= \sum_{a \in A} \left[\min_{A(x)} (\alpha) \right] \cdot \pi_t(A_t = a | X_t = x) \\ &= \left[\min_{A(x)} (\alpha) \right] \cdot \sum_{a \in A} \pi_t(A_t = a | X_t = x) \\ &= \min_{A(x)} \left[c(x,a) + \sum_{y \in X} J_{t+1}(y) \mathbb{P}(X_{t+1} = y | X_t = x, A_t = a) \right] \\ &= J_t(x), \end{aligned}$$

Por lo que se concluye la prueba de la Afirmación 2.6.

Recordando (2.12), $J(\pi, x) = C_0(\pi, x)$ para cualquier política π, entre ellas π^*, por lo que

$$J(\pi^*, x) = C_0(\pi^*, x), \tag{2.22}$$

además, haciendo uso de (2.15) con $t = 0$, $C_0(\pi^*, x) = J_0(x)$, así

$$J(\pi^*, x) = J_0(x). \tag{2.23}$$

Para demostrar la otra igualdad de (2.10), es necesario tener presente la definición

$$J^*(x) = \text{ínf}\{J(\pi, x) : \pi \in \Pi\},$$

con esto ocurre que, para toda π, $J^*(x) \leq J(\pi, x)$, en particular para π^* y por (2.23)

$$J^*(x) \leq J_0(x). \tag{2.24}$$

Para $t = 0$ en (2.13) y (2.14), $J(\pi, x) = C_0(\pi, x)$, $C_0(\pi, x) \geq J_0(x)$ respectivamente, se obtiene

$$J(\pi, x) \geq J_0(x),$$

se sigue que $J_0(x)$ es cota inferior del conjunto $\{J(\pi, x) : \pi \in \Pi\}$, entonces por definición de J^*,

$$J^*(x) \geq J_0(x), \tag{2.25}$$

haciendo uso de (2.24) y (2.25) se llega a la conclusión de que $J^*(x) = J_0(x)$, obteniendo así las igualdades deseadas

$$J^*(x) = J_0(x) = J(\pi^*, x). \tag{2.26}$$

■

2.3 Aplicaciones

Ejemplo 2.7. *Retomando el Ejemplo 1.7, la variable de acción A_t denota la cantidad pedida al principio del periodo t, en este caso la dinámica del sistema esta dado por la siguiente ecuación en diferencias:*

$$X_{t+1} = (X_t + A_t - \xi_{t+1})^+, \tag{2.27}$$

Se agrega un costo de producción, b, por lo que la función de costo queda de la siguiente manera:

$$c(X_t, A_t, \xi_t) = bA_t + h\,\text{máx}\{0, X_t + A_t - \xi_{t+1}\} + p\,\text{máx}\{0, -X_t - A_t + \xi_{t+1}\} \tag{2.28}$$

Se toma la distribución de la demanda dada en el Ejemplo 1.7 , asi como la capacidad de el almacén, M=5, por lo que $X = \{0,1,2,3,4,5\}$ y $A(x) = \{0,1,\ldots,5-x\}\,\forall x \in X$, donde X representa el espacio de estados, en este ejemplo se tomará $h = 2$ y $p = 10$. Para ejemplificar el Teorema 2.5, se toma $N = 6$, el costo terminal, $c_N(x) = 0, \forall x \in X$ y $b = 5$, de esta manera, se desea calcular:

$$J_t(x) = \min_{A(x)} \left[c(x,a) + \sum_{y \in X} J_{t+1}(y)\mathbb{P}(X_{t+1} = y | X_t = x, A_t = a) \right],\ x \in X \tag{2.29}$$

para $t = \{5,4,3,2,1,0\}$.

Siguiendo la idea del ejemplo anterior:

$$c(x,a) = b \cdot a + \left(\frac{h-p}{2}\right)(x + a - \mathbb{E}[\xi]) + \left(\frac{h+p}{2}\right)(\mathbb{E}\,[|x + a - \xi|]).$$

A continuación se muestra el Pseudocódigo 2 referente al ejemplo visto, este consta de dos partes, en la primera se observan las funciones a las cuales se les dará uso en la siguiente página donde se encuentra el código principal.

Pseudocódigo 2 Stock de un inventario (b)

```
1:  procedure FUNTION(aux,XI,X,x,a)
2:      sum=0
3:      for y in X do
4:          xi=x+a-y
5:          if 0 ≤ xi ≤ M then
6:              sum =sum+ aux[y]*XI[xi]
7:          return (sum)
8:
9:  procedure SUMA(XI,x,a,M)
10:     sum=0
11:     for s in (0,...,M-1) do
12:         sum= sum+|x+a-s|· XI[s]
13:     return(sum)
14:
15: procedure COST(XI,x,a,expxi,M)
16:     b=5
17:     h=2
18:     p=10
19:     C1=b· a
20:     C2=(h-p)/2 ·(x+a-expxi)
21:     C3=(h+p)/2 ·SUMA(XI,x,a,M)
22:     return(C1+C2+C3)
```

```
N=6
X=[0,1,...,M]
XI=[0,0.1,0.1,0.3,0.3,0.2]
exp=3.4
print("Tabla de costos: ")
tab1=[M+1,M+1]
for x in X do
    for  a in (0,...,M-x) do
        tab1[x,a]=Costo(XI,x,a,espxi,M)
print(tab1)
J=[M+1]
tab1=[ ]
for x in X do
    C=[(M+1)-x]
    for  a in (0,...,M-x) do
        C[a]=Costo(XI,x,a,espxi,M)
        posminima=argmin(C)
        J[x]=C[posminima]
print("t: ",N-1)
print("J= ",J)
for t in (N-2,...,0) do
    aux=copy(J)
    for x in X do C=[ ]
        for a in (0,...,M) do
            C[a]=Costo(XI,x,a,espxi,M)+Sumatoria(aux,XI,X,x,a)
            posminima=argmin(C)
            J[x]=C[posminima]
    print("t: ",N-1)
    print("J= ",J)
```

En la Tabla 2.1 se muestran los costos generados, dependiendo del nivel de inventario actual y los productos pedidos en acciones admisibles, recuérdese que la capacidad máxima de el almacén es 5, por ejemplo, si x=3, el conjunto de acciones admisibles es $A(x) = \{0, 1, 2\}$, el resto de las acciones no son admitidas y están representadas por "-".

x \ a	0	1	2	3	4	5
0	34	29	25.2	22.6	23.2	28.2
1	24	20.2	17.6	18.6	23.2	-
2	15.2	12.6	13.6	18.2	-	-
3	7.6	8.6	13.2	-	-	-
4	3.6	8.2	-	-	-	-
5	3.2	-	-	-	-	-

Tabla 2.1: Costos

En la etapa 5, se obtiene:

x	a=0	a=1	a=2	a=3	a=4	a=5	$J_5(x)$	s
0	34	29	25.2	22.6	23.6	28.2	22.6	3
1	24	20.2	17.6	18.6	23.2	-	17.6	2
2	15.2	12.6	13.6	18.2	-	-	12.6	1
3	7.6	8.6	13.2	-	-	-	7.6	0
4	3.6	8.2	-	-	-	-	3.6	0
5	3.2	-	-	-	-	-	3.2	0

Tabla 2.2: Valores para t=5.

Para $t = 4$:

x	a=0	a=1	a=2	a=3	a=4	a=5	$J_0(x)$	s
0	34	31.26	29.22	32.4	37.68	42.9	29.22	2
1	26.26	24.22	27.4	32.68	37.9	-	24.22	1
2	19.22	22.4	27.68	32.9	-	-	19.22	0
3	17.4	22.68	27.9	-	-	-	17.4	0
4	17.68	22.9	-	-	-	-	17.68	0
5	17.9	-	-	-	-	-	17.9	0

Tabla 2.3: Valores para t=4.

Para $t = 3$:

x	a=0	a=1	a=2	a=3	a=4	a=5	$J_0(x)$	s
0	34.	31.922	30.544	35.71	43.294	50.584	30.544	2
1	26.922	25.544	30.71	38.294	45.584	-	25.544	1
2	20.544	25.71	33.294	40.584	-	-	20.544	0
3	20.71	28.294	35.584	-	-	-	20.71	0
4	23.294	30.584	-	-	-	-	23.294	0
5	25.584	-	-	-	-	-	25.584	0

Tabla 2.4: Valores para t=3.

Para $t = 2$:

x	a=0	a=1	a=2	a=3	a=4	a=5	$J_0(x)$	s
0	34	32.054	30.809	36.372	44.552	52.536	30.809	2
1	27.054	25.809	31.372	39.552	47.536	-	25.809	1
2	20.809	26.372	34.552	42.536	-	-	20.809	0
3	21.372	29.552	37.536	-	-	-	21.372	0
4	24.552	32.536	-	-	-	-	24.552	0
5	27.536	-	-	-	-	-	27.536	0

Tabla 2.5: Valores para t=2.

Para $t = 1$:

x	a=0	a=1	a=2	a=3	a=4	a=5	$J_0(x)$	s
0	34	32.081	30.862	36.504	44.803	52.939	30.862	2
1	27.081	25.862	31.504	39.803	47.939	-	25.862	1
2	20.862	26.504	34.803	42.939	-	-	20.862	0
3	21.504	29.803	37.939	-	-	-	21.504	0
4	24.803	32.939	-	-	-	-	24.803	0
5	27.939	-	-	-	-	-	27.939	0

Tabla 2.6: Valores para t=1.

De esta manera obtenemos los valores hasta $t = 0$, con lo que se obtiene la Tabla 2.7

x	a=0	a=1	a=2	a=3	a=4	a=5	$J_0(x)$	s
0	34	32.086	30.872	36.531	44.854	53.02	30.872	2
1	27.086	25.872	31.531	39.854	48.02	-	25.872	1
2	20.872	26.531	34.854	43.02	-	-	20.872	0
3	21.531	29.854	38.02	-	-	-	21.531	0
4	24.854	33.02	-	-	-	-	24.854	0
5	28.02	-	-	-	-	-	28.02	0

Tabla 2.7: Valores para t=0.

Donde "s" representa lo que debe producirse.

De este modo, gracias al Teorema de PD 2.5 y haciendo uso de las tablas podemos concluir que la estrategia óptima es:

$$f_5(x) = \begin{cases} 0 & si \quad x \geq 3 \\ \\ 3 - x & si \quad x \leq 2 \end{cases}, f_t = \begin{cases} 0 & si \quad x \geq 2 \\ \\ 2 - x & si \quad x < 2 \end{cases}, t = 0, 1, \ldots, 4,$$

$x = 0, 1, \ldots, 5$. Para obtenerla obsérvese la Tabla 2.2, con $t = 5$, dónde los valores de x iguales a $0, 1$ y 2, y los valores de s asignados a los mismos, al ser sumados indican la cantidad de stock que debe mantenerse. Es decir cuando $x = 0, 1, 2$ deben producirse $3, 2, 1$ respectivamente de tal modo que se mantenga en 3 el stock. Mientras para $x = 3, 4, 5$, no debe producirse nada.

3 Problemas de Paro Óptimo

En este capítulo se presentan dos problemas de paro óptimo, los cuales son resueltos vía programación dinámica. Esto expone el hecho de que el presente capítulo es el punto central de la tesis, razón por la cual son aplicados aquí los conceptos antes enunciados. En la Sección 3.2 se presenta el problema de aceptar la mejor oferta, también conocido como el problema de la secretaria [3, 4]. Este es un problema conocido con una gran cantidad de variantes, que ha sido de gran interés desde los años cincuenta. Aunque su origen no es certero, la primera solución al problema estándar, el cual es objeto de estudio en ésta sección, fue dada en 1961, pero en 1963 Dynkin [4] realizó una solución al problema usando Programación Dinámica. En esta sección se presenta la solución al problema, se realizan simulaciones del mismo y se presenta el pseudocodigo para realizar dichas simulaciones. En la Sección 3.3 se presenta el problema de cubrir distintos tipos de vacantes, para el cual son necesarias diversas nociones que son introducidas en la misma sección. El problema, aunque poco conocido, es útil para identificar la diversidad de maneras que hay para trabajar la Programación Dinámica, dotando de un ejemplo dónde los estados no son triviales. De igual manera en esta sección se incluyen simulaciones y pseudocódigos para realizar las mismas.

3.1 Estudio de problemas de paro óptimo vía programación dinámica

En este capítulo se asumirá un espacio de estados contable y un espacio de acciones finito. De esta manera se supondrá que cuando una acción a es elegida, entonces se incurre en un costo esperado no negativo $C(i,a) \geq 0$. El objetivo es minimizar el costo total esperado, pues esto es equivalente al rendimiento total esperado de un problema que tiene una función de recompensa $R(i,a) = [-C(i,a)]$ la cual es no positiva.

Para alguna política π, sea

$$V_\pi(i) = \mathbb{E}^\pi \left[\sum_{n=0}^{\infty} C(X_n, a_n) \middle| X_0 = i \right]$$

Además, sea

$$V(i) = \inf_{\pi} V_{\pi}(i),$$

y la política π^* será llamada óptima si

$$V_{\pi^*}(i) = V(i), i \geq 0.$$

Es posible que $V(i)$ sea infinito, en tal caso todas las políticas serían óptimas. Este modelo solo es de interés si $V(i) < \infty$ para algunos valores de i.

Notación 3.1. *Si el proceso esta en el estado i, en el tiempo n y la acción a es elegida entonces el siguiente estado del sistema se elige de acuerdo con las probabilidades de transición $P_{ij}(a)$.*

Si X_n denota el estado del proceso al tiempo n y a_n la acción elegida en ese momento, entonces es equivalente afirmar que:

$$\mathbb{P}\{X_{n+1} = j | X_0, a_0, X_1, a_1, \ldots, X_n = i, a_n = a\} = P_{ij}(a).$$

Teorema 3.2. *La ecuación de optimalidad*

$$V(i) = \min_{a} \left[C(i,a) + \sum_{j} P_{ij}(a) V(j) \right].$$

El principal resultado teórico de la programación dinámica negativa es que la política determinada por la ecuación de optimalidad es óptima.

Teorema 3.3. *Sea f una política estacionaria definida por*

$$C(i, f(i)) + \sum_{j} \mathbb{P}_{ij}(f(i)) V(j) = \min_{a} [C(i,a) + \sum_{j} P_{ij}(a) V(j)], i \geq 0.$$

Entonces

$$V_f(i) = V(i), i \geq 0.$$

La prueba del teorema anterior se encuentra en [1]. Ahora se supondrá que cuando una acción a es elegida en el estado i, entonces un costo esperado no negativo $C(i,a) \geq 0$, es incurrido. El objetivo es minimizar el costo total esperado incurrido, esto equivale al rendimiento total esperado para un problema que tiene una función

de recompensa $R(i,a) = [-C(i,a)]$ el cual es no positivo. Para cualquier política π, sea

$$V_\pi(i) = \mathbb{E}^\pi \left[\sum_{n=0}^{\infty} C(X_n, a_n) \Big| X_0 = i \right].$$

Como $C(i,a) \geq 0,\ V_\pi(i)$ esta bien definida. Antes de continuar, se debe introducir un poco acerca de los problemas de paro óptimo, se considera el siguiente modelo: cuando se está en el estado i, es posible tomar la decisión de detener el proceso, recibir la recompensa $R(i)$ y el proceso termina, o elegir pagar $C(i)$ y continuar, entonces el siguiente estado será j con probabilidad P_{ij}. Suponiendo que $C(i)\ y\ R(i)$ son no negativos. Diciendo que el proceso va al estado ∞ cuando se elige la acción de paro, entonces, se tiene un proceso de desición de 2 acciones, donde la acción de paro es representada por la primera de ellas.
Con estas consideraciones se presentan las siguientes ecuaciones correspondientes a la probabilidad de transición y los costos que ocurren según las acciones 1 y 2, dependiendo el caso. $C(i,1) = -R(i)$, pues la recompensa terminal es representada como un costo negativo. Bajo este contexto se tiene: $P_{i,\infty}(1) = 1,\ C(i,2) = C(i),\ P_{i,j}(2) = P_{i,j},\ P_{\infty,\infty}(a) = 1,\ C(\infty,a) = 0$.

Este proceso no encaja en el marco de la programación dinámica negativa (ver definición en [1]) pues no todos los costos son no negativos. Para transformar el proceso en uno que se ajuste a las condiciones necesarias, se asume lo siguiente:

Suposición 3.4.

a) $inf_i C(i) > 0$

b) $sup_i R(i) < \infty$.

Bajo estas condiciones, sea $R := sup_i R(i)$ y considérese un problema relativo, en el cual, cuando se está en el estado i, se puede elegir parar, y recibir la recompensa terminal $R(i) - R$ (o equivalentemente pagar un costo $R - R(i) \geq 0$), o continuar pagando un costo $C(i)$ para avanzar hacia $j \geq 0$ con probabilidad P_{ij}.

Para cualquier política π, considérese $V_\pi\ y\ \bar{V}_\pi$, como las funciones de costo esperado para el problema original y el problema relativo, respectivamente.

Nótese que para cualquier política π que pare con probabilidad 1, se cumple:

$$\bar{V}_\pi(i) = V_\pi(i) + R,\ i = 0, 1, 2, \ldots$$

Además estas son las únicas políticas que es necesario considerar, debido a que, por Suposición (a), cualquier política que no se detenga con probabilidad 1, satisface:

$$V_\pi(i) = \bar{V}_\pi(i) = \infty.$$

Así, cualquier política que sea óptima para el proceso original, es óptima para el proceso relativo y viceversa. Por lo cual el proceso relativo es un proceso de desición de Markov con costos no negativos, en consecuencia, por resultado previo, existe una política óptima y la función de costo óptimo $\bar{V}$ satisface:

$$\bar{V}(i) = min\left[R - R(i), C(i) + \sum_j P_{ij}\bar{V}(j)\right],\ i \in X.$$

Además, la pólitica que elige las acciones minimizadoras es óptima por que $V(i) = \bar{V}(i) - R(i)$, véase que,

$$V(i) = min\left[-R(i), C(i) + \sum_j P_{ij}V(j)\right],\ i \in X \tag{3.1}$$

y la política así determinada es óptima.
Sea $V_0(i) = -R(i)$, y para $n \geq 1$,

$$V_n(i) = min\left[-R(i), C(i) + \sum_j P_{ij}V_{n-1}(j)\right]\ i \in X. \tag{3.2}$$

Recuérdese que $V_n(i)$ representa el costo total minimo esperado en el que se incurre, si se comienza en el estado i y es permitido un máximo de n etapas antes de detenerse, desde esta interpretación se puede ver que $V_n(i) \geq V_{n+1}(i) \geq V(i)$, así

$$\lim_{n\to\infty} V_n(i) \geq V(i) \tag{3.3}$$

y se dice que el proceso es estable si $\lim_{n\to\infty} V_n(i) = V(i)$.

Proposición 3.5. *Bajo las condiciones*

a) $C := \inf_i C(i) > 0$,

b) $R := \sup_i R(i) < +\infty$,

Se tiene que para todo n y toda i,

$$V_n(i) - V(i) \leq \frac{(R-c)[R-R(i)]}{(n+1)\cdot C}.$$

Demostración. Sea f una política óptima y sea T el tiempo aleatorio en el cual f se detiene. Además, sea f_n la política que elige la misma acción que f al tiempo $0, 1, \ldots, n-1$ pero que se detiene al tiempo n (si no se ha detenido previamente), donde $\mathcal{S}$ denota el costo total,

$$V(i) = V_f(i) = \mathbb{E}_i^f(\mathcal{S}|T \leq n) \cdot \mathbb{P}(T \leq n) + \mathbb{E}_i^f(\mathcal{S}|T > n) \cdot \mathbb{P}(T > n) \tag{3.4}$$

y

$$V_n(i) \leq V_{f_n}(i) = \mathbb{E}_i^f(\mathcal{S}|T \leq n) \cdot \mathbb{P}(T \leq n) + \mathbb{E}_i^{f_n}(\mathcal{S}|T > n) \cdot \mathbb{P}(T > n).$$

De esta manera

$$V_n(i) - V(i) \leq \left[\mathbb{E}_i^{f_n}(\mathcal{S}|T > n) - \mathbb{E}_i^f(\mathcal{S}|T > n)\right] \cdot \mathbb{P}(T > n) \leq (R - C) \cdot \mathbb{P}(T > n)$$

Para obtener una cota en $\mathbb{P}(T > n)$, haciendo uso de (3.4) se obtiene:

$$\begin{aligned} -R(i) \geq V(i) &\geq -R \cdot \mathbb{P}(T \leq n) + [-R + (n+1) \cdot C] \cdot \mathbb{P}(T > n) \\ &= -R + (n+1) \cdot C \cdot \mathbb{P}(T > n) \end{aligned}$$

entonces

$$\mathbb{P}(T > n) \leq \frac{R - R(i)}{(n+1) \cdot C}$$

y el resultado se sigue. ■

Observación 3.6. *Bajo las condiciones de la proposición anterior se cumple*

$$V_n(i) - V(i) \leq \frac{(R-C)(R-R(i))}{(n+1)C},$$

haciendo $n \longrightarrow \infty$ en ambos lados de la desigualdad, se obtiene:

$$\lim_{n\to\infty} [V_n(i) - V(i)] \leq \lim_{n\to\infty} \frac{(R-C)(R-R(i))}{(n+1)C},$$

notar que $\frac{(R-C)(R-R(i))}{C}$ no depende de n, entonces:

$$\begin{aligned} \lim_{n\to\infty} V_n(i) - V(i) &\leq 0 \\ \lim_{n\to\infty} V_n(i) &\leq V(i) \end{aligned}$$

y haciendo uso de (3.3) $\lim_{n\to\infty} V_n(i) = V(i)$, es decir, cuando se cumplen las condiciones $a)$ y $b)$ el proceso es estable.

Para ver que no todos los problemas de paro son estables considérese el siguiente ejemplo.

Ejemplo 3.7. *Problema de paro no estable.*

Sea el espacio de estados, el conjunto de todos los enteros y sea: $C(i) \equiv 0, R(i) = i, P_{i,i+1} = \frac{1}{2} = P_{i,i-1}$. *Luego* $V_0(i) = -i$, *y puede probarse que* $V_n(i) = -i$, *sin embargo, hasta que el problema se detenga, el estado cambia de acuerdo a una caminata aleatoria simétrica, pero tal cadena de Markov es recurrente nula y por lo tanto, se sabe que, con probabilidad* 1, *cualquier estado* N *dado eventualmente será alcanzado. Por lo que, si se comienza en el estado* i, *se puede garantizar una recompensa final de* N, *usando la política de detenerse cuando el proceso entra a* N. *Como es cierto para toda* N, *se puede notar que* $V(i) = -\infty$. *Así, este problema de paro, el cual no satisface* $a)$ *y* $b)$, *no es estable.*

La siguiente teoría sera utilizada en la Sección 3.3.

Sea

$$B = \left\{ i : -R(i) \leq C(i) - \sum_{j=0}^{\infty} P_{ij} R(j) \right\} = \left\{ i : R(i) \geq \sum_{j=0}^{\infty} P_{ij} R(j) - C(i) \right\}.$$

La política que se detiene la primera vez que el proceso entra a un estado en B es llamada, la política de anticipación de una etapa (one-stage look-ahead policy).

Definición 3.8. *Un conjunto de estados* B, *es cerrado si para cualesquiera* $i \in B$ *y* $j \notin B$ *se satisface que* $P_{ij} = 0$.

Se probará un resultado tal que si B es un conjunto cerrado de estados, entonces, bajo el supuesto de estabilidad y la política de una etapa es óptima.

Teorema 3.9. *Si el proceso es estable y si* $P_{ij} = 0$ *para* $i \in B$, $j \notin B$, *entonces la política óptima se detiene en* i *si y solo si* $i \in B$.

Demostración. Se mostrará por inducción sobre n que $V_n(i) = -R(i)$, $\forall i \in B \, y \, \forall n \geq 0$. Por (3.2), $V_0(i) = -R(i)$. Suponiendo que se cumple para $n-1$, entonces para $i \in B$,

$$
\begin{aligned}
V_n(i) &= \text{mín}\left[-R(i); C(i) + \sum_{j=0}^{\infty} P_{ij} V_{n-1}(j)\right] \\
&= \text{mín}\left[-R(i); C(i) + \sum_{j\in B} P_{ij} V_{n-1}(j)\right] \text{ (pues B es cerrado)} \\
&= \text{mín}\left[-R(i); C(i) - \sum_{j\in B} P_{ij} R(j)\right] \text{ (hip. de inducción)} \\
&= \text{mín}\left[-R(i); C(i) - \sum_{j=0}^{\infty} P_{ij} R(j)\right].
\end{aligned}
$$

Como $i \in B$, $-R(i) \leq C(i) - \sum_{j=0}^{\infty} P_{ij} R(j)$, de esta manera $V_n(i) = -R(i)$, $\forall i \in B$ y $\forall n \geq 0$. Utilizando hipótesis de estabilidad,

$$
\begin{aligned}
V(i) &= \lim_{n\to\infty} V_n(i) \\
&= \lim_{n\to\infty} -R(i) \\
&= -R(i),
\end{aligned}
$$

se obtiene $V(i) = -R(i)$, $i \in B$. Ahora para $i \notin B$ la política que continua por exactamente una etapa y se detiene, tiene un costo esperado

$$
C(i) - \sum_{j=0}^{\infty} P_{ij} R(j),
$$

el cual es estrictamente menor que $-R(i)$ (pues $i \notin B$), es decir,

$$
C(i) - \sum_{j=0}^{\infty} P_{ij} R(j) < -R(i). \tag{3.5}
$$

Por otro lado por (3.2), para cada j, $V_n(j) \leq -R(j)$, así

$$
\lim_{n\to\infty} V_n(j) \leq \lim_{n\to\infty} -R(j).
$$

Dado que el proceso es estable $V(j) \leq -R(j)$, con lo cual se obtiene:

$$
\begin{aligned}
\sum_{j=0}^{\infty} P_{ij} V(j) &\leq -\sum_{j=0}^{\infty} P_{ij} R(j) \\
C(i) + \sum_{j=0}^{\infty} P_{ij} V(j) &\leq C(i) - \sum_{j=0}^{\infty} P_{ij} R(j),
\end{aligned}
$$

haciendo uso de (3.1) y (3.5), implica $V(i) < -R(i)$.

En resumen,

$$V(i) = -R(i),\ i \in B$$
$$V(i) < -R(i),\ i \notin B,$$

es decir, si $i \in B$ la acción sugerida por la política es detenerse, en caso contrario la acción es continuar. ■

3.2 Aceptando la mejor oferta

Suponga que se presentan n ofertas en orden secuencial, con $n \geq 1$, donde los $n!$ órdenes son igualmente probables, la única información con la que se cuenta en cada etapa es el rango relativo de la oferta actual en comparación con las anteriores. Al momento que se presenta una oferta, se puede decidir si se rechaza, y se continua el proceso, o es aceptada, y el proceso termina. El objetivo es maximizar la probabilidad de seleccionar la mejor oferta.

Se definen:

- i es una oferta candidato, si su valor es mayor o igual que el de las ofertas anteriores.
- $V(i) \equiv$ Probabilidad de elegir la mejor oferta en la posición i.
- $P(i) \equiv$ Probabilidad de que, si la i-ésima oferta es aceptada ésta sea la mejor de todas.
- $H(i) \equiv$ Probabilidad de obtener la mejor oferta cuando la i-ésima oferta es rechazada.

De acuerdo a la ecuación de programación dinámica (2.8) se obtiene:

- $V(i) = \max[P(i), H(i)]$
- $P(i) = \mathbb{P}(\text{la oferta } i\text{-ésima es la mejor de todas} | \text{ la oferta } i\text{-ésima es la mejor de las primeras } i) = \frac{\frac{1}{n}}{\frac{1}{i}} = \frac{i}{n}$.
- $H(i) = \frac{1}{i+1}V(i+1) + \frac{i}{i+1}H(i+1)$.

Es posible ver las igualdades anteriores, ya que, si la oferta i es rechazada, la oferta $i+1$ puede ser aceptada o rechazada.

La probabilidad de que la oferta $i+1$ sea mayor que las anteriores es $\frac{1}{i+1}$ y la probabilidad de ganar dado ese evento es, por definición, $V(i+1)$.

La probabilidad de que la oferta $i+1$ no sea mayor que sus antecesoras es

$$1 - \frac{1}{i+1} = \frac{i}{i+1}$$

pero, dado que no es mayor que las anteriores, obtenemos la mejor oferta con probabilidad 0 si la aceptamos, por lo que se debe rechazar la oferta $i+1$ y la probabilidad de obtener la mejor oferta dado que fue rechazada la $i+1$ es por definición $H(i+1)$.

V y H son ecuaciones recursivas y para determinarlas hay una condición inicial, $H(n) = 0$ pues la probabilidad de obtener la mejor oferta dado que rechazamos la última y $V(n) = 1$, por como esta definida se debe elegir el máximo entre $1\ y\ 0$.

Para ver la forma recursiva de H, se evalúa en la penúltima etapa:

$$\begin{aligned} H(n-1) &= \frac{1}{n} \cdot V(n) + \frac{n-1}{n} \cdot H(n) \\ &= \frac{1}{n} \\ &= \left(\frac{n-1}{n}\right)\left(\frac{1}{n-1}\right). \end{aligned}$$

Entonces,

$$\begin{aligned} V(n-1) &= \text{máx}\left\{\frac{n-1}{n}, \frac{n-1}{n} \cdot \frac{1}{n-1}\right\} \\ &= \frac{n-1}{n} \cdot \text{máx}\left\{1, \frac{1}{n-1}\right\} \\ &= \frac{n-1}{n} \end{aligned}$$

Para $n-2$:

$$\begin{aligned} H(n-2) &= \frac{1}{n-1} \cdot V(n-1) + \frac{n-2}{n-1} \cdot H(n-1) \\ &= \left(\frac{1}{n-1}\right)\left(\frac{n-1}{n}\right) + \frac{n-2}{n-1}\left(\frac{1}{n}\right) \\ &= \left(\frac{n-2}{n}\right)\left(\frac{1}{n-2}\right) + \left(\frac{n-2}{n}\right)\left(\frac{1}{n-1}\right) \\ &= \frac{n-2}{n}\left(\frac{1}{n-2} + \frac{1}{n-1}\right). \end{aligned}$$

Lo cual implica

$$\begin{aligned} V(n-2) &= \max\left\{\frac{n-2}{n}, \frac{n-2}{n}\left(\frac{1}{n-2}+\frac{1}{n-1}\right)\right\} \\ &= \frac{n-2}{n}\cdot \max\left\{1, \frac{1}{n-2}+\frac{1}{n-1}\right\}. \end{aligned}$$

De manera general se obtiene

$$\begin{aligned} H(i) &= \frac{i}{n}\cdot\sum_{m=i}^{n-1}\frac{1}{m} \\ V(i) &= \frac{i}{n}\cdot \max\left\{\ 1, \sum_{m=i}^{n-1}\frac{1}{m}\right\}. \end{aligned}$$

Observación 3.10. *Cuando* $\sum_{m=1}^{n-1}\frac{1}{m}\geq 1$*, las ofertas son rechazadas, pues* $V(i)=H(i)$ *y cuando* $\sum_{m=1}^{n-1}\frac{1}{m}\leq 1$ *se considera la opción de aceptar la oferta. Además* $\sum_{m=i}^{n-1}\frac{1}{m}$ *es una función decreciente, siempre que* $n>2$*,*

$$\sum_{m=1}^{n-1}\frac{1}{m}>1 \quad y \quad \sum_{m=n-1}^{n-1}\frac{1}{m}<1.$$

De aquí se deriva la existencia de $j<n$ *tal que:*

$$\sum_{m=j+1}^{n-1}\frac{1}{m}<1\leq\sum_{m=j}^{n-1}\frac{1}{m}.$$

H es una función decreciente después de pasar j por lo cual, después de este se debe aceptar la oferta con mayor o igual valor al máximo de las anteriores.

Recuérdese que el estudio se hace sobre las ofertas candidato, i.e., lo que conviene es rechazar las primeras ofertas candidato hasta la oferta j y como H decrece, se acepta la siguiente oferta candidato. Resta encontrar el momento j,

$$\begin{aligned} H(j) &= \frac{j}{n} \cdot \sum_{m=j}^{n-1} \frac{1}{m} && (3.6) \\ &\approx \frac{j}{n} \cdot \int_{j}^{n-1} \frac{1}{x} dx \\ &= \frac{j}{n} \cdot log\left(\frac{n-1}{j}\right) \\ &\approx \frac{j}{n} \cdot log\left(\frac{n}{j}\right) && (3.7) \end{aligned}$$

En (3.6) la suma se aproxima por sumas de Riemman a una integral, mientras que en (3.7) es posible hacer la aproximación ya que la función logaritmo crece con distancias pequeñas.

Por otro lado, si

$$g(x) = \frac{x}{n} \cdot log\left(\frac{n}{x}\right),$$

se tiene que:

$$g'(x) = \frac{1}{n} \cdot log\left(\frac{n}{x}\right) - \frac{1}{n},$$

igualando a cero, $log\left(\frac{n}{x}\right) = 1$, entonces $x = \frac{n}{e}$.

La estrategia óptima es rechazar las primeras $\frac{n}{e}$ ofertas y después aceptar la primer oferta candidato que aparezca, la probabilidad de obtener la mejor de todas será $\frac{1}{e}$, pues $g\left(\frac{n}{e}\right) = \frac{1}{e}$.

Recuérdese que, g aproxima a H, i.e., la probabilidad de obtener la mejor oferta dado que se dejan pasar $\frac{n}{e}$ es $\frac{1}{e}$.

A continuación se muestra el Pseudocódigo 3 para realizar la simulación del problema, considerando N como el número total de ofertas presentadas.

Pseudocódigo 3 Aceptando la mejor oferta

```
N = 100
j = int(N/e)
best = 0
aux = 0
for k in (1, 100) do
    oferta[k] =randint(1, 100)
    if oferta[k] > best then
        best = oferta[k]
    if k > j and aux == 0 then
        if oferta[k] ≥ best then
            chosen = oferta[k]
            write("La oferta elegida es: ")
            write(chosen)
            aux = 1
write("La mejor oferta es: ")
write(best)
```

Realizando simulaciones en Python, se generan cien ofertas en orden aleatorio, dentro de un rango de 0 a 100, de las cuales una oferta es elegida siguiendo la estrategia anterior, sin embargo, en algunos casos, puede presentarse una oferta mas alta, después de ya haber elegido alguna otra. El número de ofertas rechazadas para este ejemplo es de $\frac{100}{e} = 36$, en la Figura 3.1 se muestra como la oferta elegida fue de 99, teniendo que la mejor oferta fue de 100.

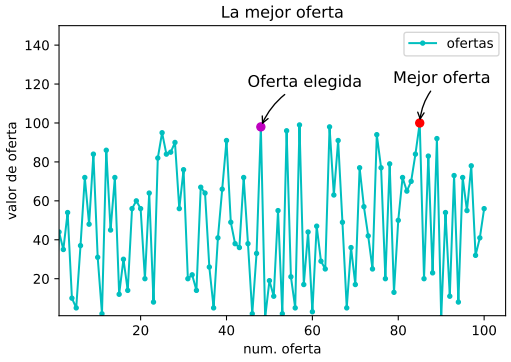

Figura 3.1

Se realizaron cincuenta simulaciones de cien ofertas cada una. En cada simulación se comparó la oferta elegida con la mejor de dicha simulación, si estas coinciden se dice que la diferencia entre ellas es cero, de otra manera, se grafica la discrepancia entre ellas y al final de esto se obtiene que el porcentaje de coincidencia fue de $62\,\%$, lo cual se muestra en la Figura 3.2

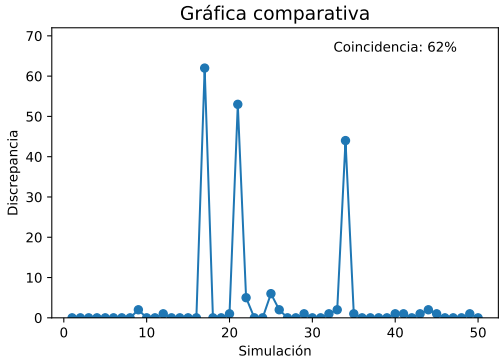

Figura 3.2

Seguido de esto, para las siguientes simulaciones, se conserva la idea anterior, donde la diferencia es el número de ofertas que son rechazadas antes de elegir la oferta candidato, en la primera gráfica de izquierda a derecha de la Figura 3.3 se rechazan las primeras 20 ofertas, donde el porcentaje de coincidencia fue de $40\,\%$, para la segunda gráfica de la Figura 3.3 se rechazan las primeras 80 ofertas y se obtiene un $26\,\%$ de coincidencia.

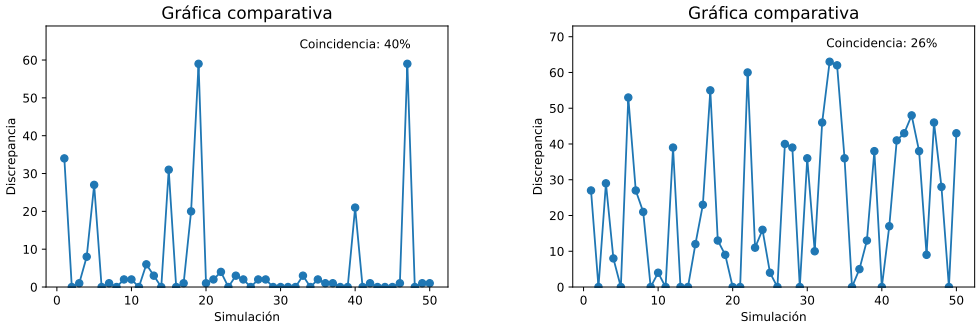

Figura 3.3

Puede observarse de las simulaciones como el porcentaje de coincidencia entre la mejor oferta y la oferta elegida de una simulación es claramente superior al seguir la política dada por la solución del problema, comprobando así el hecho ya conocido de que esta es la política óptima. Más aún, la Figura 3.2 y la Figura 3.3 ponen en evidencia que la discrepancia entre oferta elegida y mejor oferta de una simulación es mucho mayor al seguir una política diferente a la óptima, suceso que refuta más la teoría. Se ve de esta forma, como resolver un problema vía programación dinámica y como se verifica empíricamente.

3.3 Cubriendo distintos tipos de vacantes

Un entrevistador es contratado por una empresa, donde hay N tipos de puestos disponibles, cada vez que se entrevista a un candidato, independientemente de los entrevistados con anterioridad, aplicará para el puesto i con probabilidad P_i para $i = 1, \ldots, N$. En cada etapa la empresa puede decidir pagar un costo C al entrevistador, al cual solo le es posible entrevistar a lo más a un candidato en dicha etapa o decidir no hacerlo. Si decide pagar el costo, puede llegar un candidato en busca de empleo con probabilidad α, si decide no continuar pagando al entrevistador, la empresa recibe la recompensa terminal $R(n)$ generada por los n tipos de puestos cubiertos. Ahora se determinará la política óptima.
El estado en cualquier momento es el conjunto X de los distintos puestos que han sido cubiertos. La política de anticipación de una etapa se detendría en X si $X \in B$, $|X|$ es la cardinalidad de X (el número de elementos en X).

$$\begin{aligned} B &= \left\{\mathbf{X} : \mathbf{R}(|\mathbf{X}|) \geq \mathbb{E}_{|\mathbf{X}|}[\mathbf{R}(\mathbf{X_1})] - \mathbf{C}\right\} \\ &= \left\{X : R(|X|) \geq \alpha \sum_{i \notin X} P_i R(|X|+1) + \alpha \sum_{i \in X} P_i R(|X|) + (1-\alpha) R(|X|) - C\right\}. \end{aligned}$$

Se denota E al evento donde un candidato es entrevistado y ese puesto ya esta ocupado, es decir, ese puesto se encuentra en X y E_i donde un candidato que aplica para el puesto i es entrevistado, de esta manera:

$$\begin{aligned} \sum_{i \in X} P_i &= \mathbb{P}(\bigcup_{i \in X} E_i) \\ &= \mathbb{P}(E) \\ &= 1 - \mathbb{P}(E^c) \\ &= 1 - \mathbb{P}(\bigcup_{i \notin X} E_i) \\ &= 1 - \sum_{i \notin X} P_i, \end{aligned}$$

entonces

$$\begin{aligned} B &= \left\{X : R(|X|) \geq \alpha \sum_{i \notin X} P_i R(|X|+1) + \alpha R(|X|)\left[1 - \sum_{i \notin X} P_i\right] + R(|X|) - \alpha R(|X|) - C\right\} \\ &= \left\{X : R(|X| \geq \alpha \sum_{i \notin X} P_i R(|X|+1) - \alpha R(|X|) \sum_{i \notin X} P_i + R(|X|) - C\right\}. \end{aligned}$$

Es posible reescribir a B como:

$$\begin{aligned} B &= \left\{ X : C \geq R(|X|+1) - R(|X|) \left[\alpha \sum_{i \notin X} P_i \right] \right\} \\ &= \left\{ X : \ R(|X|+1) - R(|X|) \leq \frac{C}{\alpha \overline{P}(X)} \right\}, \end{aligned}$$

donde $\overline{P}(X) = \sum_{i \notin X} P_i$.

Ahora, suponiendo que $R(n)$ es una función cóncava, se probará que $R(n+1) - R(n)$ decrece en n, sea $g(n) = R(n+1) - R(n)$, quiere probarse que g es decreciente, donde las siguientes desigualdades son equivalentes,

$$\begin{aligned} g(n+1) &\leq g(n) \\ R(n+2) - R(n+1) &\leq R(n+1) - R(n) \\ R(n+2) + R(n) &\leq 2R(n+1), \end{aligned}$$

por suposición R es cóncava, es decir, para todo $x,\ y \in dom(R)$ y $t \in [0,1]$,

$$R(tx + (1-t)y) \geq tR(x) + (1-t)R(y).$$

Sean $x = n,\ y = n+1$ y $t = \frac{1}{2}$,

$$\begin{aligned} R(n+1) &= R(\frac{1}{2}n + \frac{1}{2}(n+2)) \\ &\geq \frac{1}{2}R(n) + \frac{1}{2}R(n+2), \end{aligned}$$

entonces

$$R(n) + R(n+2) \leq 2R(n+1),$$

de esta manera g es decreciente.

Para demostrar que la política es óptima se hará uso del Teorema 3.9, para lo cual es necesario probar el siguiente lema.

Lema 3.11. *El proceso es estable y el conjunto B es cerrado.*

Demostración. La condición $a) inf_{|X|} C(|X|) > 0$ se cumple, ya que para $i = 1, \ldots, N, C(i) = C$, positiva, de esta manera $inf_{|X|} C(|X|) = C$.
La condición $b) sup_{|X|} R(|X|) < \infty$ se cumple, ya que al existir una cantidad finita de recompensas, $sup_{|X|} R(|X|) = \max_{|X|} R(|X|)$. Así, por la Observación 3.6 el proceso es estable. Resta ver que B es un conjunto cerrado, para ello, sea $X \in B,\ X' \notin B$,

se quiere probar que $P_{XX'} = 0$, suponiendo que esto no ocurre hay que notar que $X \subseteq X'$ o $X \not\subseteq X'$. Si $X \not\subseteq X'$, entonces existe $i \in X$ tal que $i \notin X'$, en consecuencia $P_{XX'} = 0$, esto debido a que si en el conjunto X se tiene el puesto del tipo i, por la naturaleza del problema, no es posible despedir empleados, es decir, desocupar puestos, por esto no es posible trasladarse de X *a* X'.

Entonces $X \subseteq X'$, más aún, $X \subsetneq X'$ puesto que $X \in B$ mientras que el otro no. Dado que $X \in B$, cumple $R(|X|+1) - R(|X|) \leq \dfrac{C}{\alpha \overline{P}(X)}$, se sabe que $R(|X|+1) - R(|X|)$ decrece y por la contención $|X| < |X'|$, por consiguiente

$$R(|X'|+1) - R(|X'|) \leq R(|X|+1) - R(|X|). \tag{3.8}$$

Cabe señalar que X' tiene a lo más N elementos y haciendo uso de la contención propia X tiene a lo más $N-1$ elementos, por tanto, existe $j \notin X$ entonces $\overline{P}(X) \geq P_j$, en otras palabras, la suma de las probabilidades de entrevistar a un candidato que aplique para un puesto que no este en X será mayor igual a la probabilidad de entrevistar a un candidato que aplique para un puesto disponible. Si $\overline{P}(X') = 0$,

$$\alpha \overline{P}(X') R(|X'|+1) + \left[1 - \alpha \overline{P}(X')\right] R(|X'|) - C = R(|X'|) - C \leq R(|X'|),$$

lo cual es la condición para pertenecer a B, pero por hipótesis $X' \notin B$, así se afirma que $\overline{P}(X) \neq 0$, $\overline{P}(X') \neq 0$. Recordando, $\overline{P}(X) = \sum_{i \notin X} P_i = \sum_{i \in X^c} P_i$, de igual manera para X', además $X'^c \subset X^c$, esto es una consecuencia de $X \subset X'$, de ahí que:

$$\begin{aligned}
\overline{P}(X) &= \sum_{i \notin X} P_i \\
&= \sum_{i \in X^c} P_i \\
&= \sum_{i \in X'^c} P_i + R \\
&= \sum_{i \notin X'} P_i + R \\
&= \overline{P}(X') + R
\end{aligned}$$

entonces

$$\overline{P}(X) - \overline{P}(X') = R,$$

con $R = \sum_{i \in X^c \setminus X'^c} P_i$ y $R \geq 0$ pues es suma de probabilidades, esto implica que

$$\overline{P}(X) \geq \overline{P}(X'),$$

entonces

$$\frac{1}{\overline{P}(X)} < \frac{1}{\overline{P}(X')},$$

en síntesis

$$\frac{C}{\alpha\overline{P}(X)} < \frac{C}{\alpha\overline{P}(X')}. \tag{3.9}$$

Haciendo uso de (3.8) y (3.9) se obtiene

$$R(|X'|+1) - R(|X'|) < \frac{C}{\alpha\overline{P}(X')},$$

lo cual es una contradicción pues $X' \notin B$. En conclusión B es cerrado. ■

Por consecuencia del lema anterior, la política es óptima cuando R es cóncava.

Al igual que en la sección anterior, se presenta el Pseudocódigo 4 para realizar una simulación de este problema, usando como función cóncava a: $10 - (\frac{x}{2} - 3)^2$, considerando 50 puestos diferentes y con una probabilidad de que alguien llegue en busca de empleo de 0.7.

Pseudocódigo 4 Cubriendo distintos tipos de vacantes

```
1:  procedure REWARD(x)
2:      y = 50 − (x/4 − 7)^2
3:      return y
4:
5:  procedure STOP(type,X,C,alpha,sum1)
6:      funtion1 =REWARD(|X|)
7:      funtion2 =REWARD(|X+1|)
8:      Pbar =suma(type)-sum1
9:      if (Pbar ≠ 0) then
10:         if (funtion2 − funtion1 ≤ C/(α·Pbar)) then
11:             return 1
12:         else
13:             return 0
14:     else
15:         return 1
16:
17: procedure FIND(u,alpha,N)
18:     for k in(1,N) do
19:         if ((k−1)·alpha/N ≤ u < k·alpha/N) then
20:             return k
21:
```

```
1: N = 50
2: alpha = 0.7
3: C = 0.5                                  ▷ Se define un costo para continuar
4: X = [ ]
5: sum1 = 0
6: i=N+1
7: flag = 0
8: for k in (1, N) do
9:     type[k] = 1/N
10:
11: while (k < 80 and flag == 0) do
12:     u = random(0, 1)
13:     if (u ≤ alpha) then
14:         i=FIND(u,alpha,N)
15:         if (i ∉ X) then
16:             add i to X
17:             sum1 = sum1 + type[i]
18:             aux=STOP(type, X, C, alpha, sum1)
19:             if (aux == 1 and flag == 0) then
20:                 flag = 1
21:                 chose = |X|
22:                 safe.k = k
23:                 rew =REWARD(|X|)-C
24:                 write("La cardinalidad de X es:")
25:                 write(|X|)
26:                 write("Num. de iteraciones:")
27:                 write(k)
28:                 write(Recompensa: ")
29:                 write(REWARD(|X|)-C)
30:     C = C + 0.5
```

Haciendo uso del pseudocódigo anterior, se realizaron simulaciones en Python, con $N = 50$ puestos disponibles en alguna empresa, una distribución uniforme para los diferentes tipos de empleos, una probabilidad $\alpha = 0.7$ de que un candidato llegue en busca de empleo y un costo por etapa $C = 0.5$.

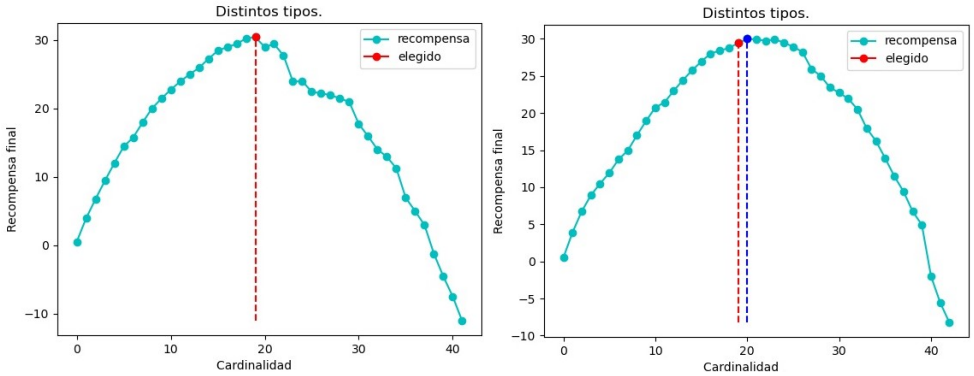

Figura 3.4

En la Figura 3.4 se muestra en color azul claro las diferentes recompensas finales obtenidas cuando la cardinalidad de X varía, pues cada que X crece, un puesto mas ha sido cubierto. La empresa genera una ganancia dependiendo del número de puestos cubiertos, a esta le restamos el costo acumulado, el cual es $k(0.5)$ donde k es el número de etapas que han transcurrido desde el inicio del proceso hasta el puesto cubierto mas reciente. La linea punteada roja, representa nuestra elección basada en la política vista previamente, la linea punteada azul indica la mayor recompensa de toda la simulación, cuando ambas lineas coinciden solo es visible la linea roja.

En estás simulaciones puede observarse como la recompensa terminal está íntimamente relacionada con la función de recompensa, obteniendo con ello una visión más clara del porque la función de recompensa debe ser cóncava para que la estrategia sea óptima. Para realizar dichas simulaciones, la probabilidad P_i fue elegida con una variable uniforme, sin embargo, en los códigos utilizados para este fin, es posible utilizar otra distribución.

Conclusión

En este trabajo de tesis se abordó la teoría de procesos de decisión de Markov, se trabajaron problemas a tiempo discreto, con horizonte finito. El criterio de rendimiento utilizado para evaluar la calidad de las políticas fue el costo total esperado, este es presentado en el Capítulo 1.

La teoría de procesos de decisión de Markov fue presentada en el Capítulo 2, además se provee una técnica de solución para los problemas de control óptimo que consisten en encontrar una política que optimice el criterio de rendimiento, lo cual se ejemplificó en el ejemplo de inventario, donde las estrategias óptimas fueron caracterizadas usando el teorema de la programación dinámica.

En el capitulo 3, se presentaron los problemas de paro óptimo, los cuales son resueltos vía programación dinámica, asimismo se presentaron dos ejemplos de dichos problemas, en el primer ejemplo, Aceptando la mejor oferta, se mostró la manera para llegar a la política óptima y se enunció dicha política, haciendo uso del pseudocódigo se realizaron simulaciones para mostrar de manera numérica los resultados, lo cual permite resolver el problema para un número mayor de estados, el mismo procedimiento se siguió para el segundo problema.

Algunos problemas cuyo análisis posterior puede ser realizado, a consecuencia de este trabajo son los siguientes:

i. Plantear y resolver el ejemplo de vacantes para un número mayor de acciones, estados y periodos.

ii. Estudiar otros métodos de solución a parte de la programación dinámica.

iii. Investigar acerca de un conjunto de datos para aplicar la teoría desarrollada.

iv. Modelar usando problemas de decisión de Markov problemas de aplicación cotidiana y dar una solución.

Anexos

Anexo 1

```
import numpy as np
from random import random

def matriz(M,V,x):
    for i in np.arange(0,6,1):
        M[i,x]=V[i,0]
    return M

def Costo(XI,x,espxi,exp2):
    h=3
    p=5

    C1=((h-p)/2)*(x-espxi)
    C2=((h+p)/2)*exp2[x,0]
    return (C1+C2)

def Sumatoria(aux,XI,X,x):

    sumatoria=0
    for y in X:
        a=x-y
        if(0<=a<=(len(XI)-1)):
            sumatoria =sumatoria+ aux[y,0]*XI[a,0]
    return sumatoria

M=5
X=np.arange(0,M+1,1) #espacio de estados
print("El espacio de estados es:")
print(X)
```

```
N=int(input(" N ?␣"))#numero de etapas

XI=np.array([[0],[0.1],[0.1],[0.3],[0.3],[0.2]])
espxi=3.4
exp2=np.array([[3.4],[2.4],[1.6],[1.0],[1.0],[1.6]])

V=np.zeros((M+1,1)) #representa V_0
for x in X:
    V[x,0]=Costo(XI,x,espxi,exp2)

aux=np.zeros((M+1,1))
M=np.zeros((M+1,N))
M=matriz(M,V,0)

for t in np.arange(1,N,1):
     for x in X:
        V[x,0]=(Sumatoria(aux,XI,X,x)+Costo(XI,x,espxi,exp2))
    M=matriz(M,V,t)#organiza los V_t en una matriz
    print("V␣es:␣")
    print(V)
print(M)
```

Anexo 2

```
import numpy as np
from random import randint
import math
N=100#de ofertas
ele=0
best=0
ban=0

j=int(N/math.exp(1))
pre=np.zeros((1,j))
post=np.zeros((1,N-j))

#hay 100 ofertas pues arange llega hasta N-1
for k in np.arange(0,N,1):

    if(k<j):
        #se genera la oferta i de 0 a 100
        pre[0,k]=randint(1,100)
    else:
        post[0,k-j]=randint(1,100)

#determina la mayor oferta presentada
if(int(np.amax(pre))>int(np.amax(post))):
    best=int(np.amax(pre))
else:
    best=int(np.amax(post))

#para determinar que oferta que se elige
#solo es necesario recorrer post,
#pues la mayor oferta de pre es la oferta
#candidato hasta ese momento
bigpre=int(np.amax(pre))
print(bigpre)
elegida=0
for k in np.arange(0,N-j,1):
    if(k==(N-j-1)):
        elegida=post[0,k]

    if(post[0,k]>=bigpre):
```

```
            elegida=post[0,k]
            break

print("pre=",pre)
print("post= ",post)
print("the best= ",best)
print("elegida= ",elegida)
```

Anexo 3

```
import numpy as np
import random
import math
import matplotlib.pyplot as plt

def recompensa(x):
    y=((x/4)-7)**2
    return (50-y)

def paro(pi,S,alpha,sumS):
    C=0.5#C es fijo

    fun1=recompensa(len(S))
    fun2=recompensa(len(S)+1)
    Pbarra=np.sum(pi)-sumS
    num=alpha*Pbarra
    beta=fun2-fun1

    if(Pbarra!=0):
        if (beta<=(C/num)):
            return 1#debe de pararse
        else:
            return 0
    else:
        return 1# pues si la suma es cero debe detenerse

def buscar(u,alpha,N):
    for k in np.arange(1,N+1):
        if((u>=((k-1)*alpha)/N)and (u<(k*alpha/N))):
            return k


N=50
X=np.arange(1,N+1,1)
print("ESPACIO DE ESTADOS\n")
print(X)#espacio de estados



C=0
alpha=0.7
```

```
print("alpha=␣")
print(alpha)

pi=np.zeros((1,N+1))#hacemos 31 espacios para dejar el primero vacio
#se tienen tipos del 1 hasta N
#la primera entrada es con k=0

for k in np.arange(1,N+1):
    pi[0,k] = 1/N

S=[]
rew=[]#servira para graficar
rew.append(recompensa(len(S))-0.5)
#la recompensa cuando la cardinalidad es cero

aux=0
ban=0
#esta suma acumulara cada probabilidad del tipo de empleo que fue cubierto
sumS=0
i=N+1
elegido=0
safek=0
tope=101
ganancia=0

k=0
while(k<tope):
    C=C+0.5
    u=random.random()

    #simulacion bernoulli
    if(u<=alpha):
        i=buscar(u,alpha,N)

        if(S.count(i)==0):
            S.append(i)
            sumS=sumS+pi[0,i]
            aux=paro(pi,S,alpha,sumS)

            if(aux==1 and ban==0):
                ban=1
                elegido=len(S)
```

```
                    print(S)
                    sakek=k
                    recom=recompensa(len(S))-C

                rew.append(recompensa(len(S))-C)

    k=k+1

print("elegido= ",elegido)
print("recom: ",recom)
print("rew18: ",rew[18])
print("rew19: ",rew[19])
print("rew20: ",rew[20])
ejeX=np.arange(0,len(S)+1,1)

plt.plot(ejeX,rew,'c-o',label='recompensa')
plt.plot(elegido,recom,'r-o',label='elegido')
plt.plot([elegido,elegido],[np.min(rew),recom],'r--')

if(np.max(rew)>recom):
    plt.plot(rew.index(np.max(rew)),np.max(rew),'b-o')
    plt.plot([rew.index(np.max(rew)),rew.index(np.max(rew))],[np.min(rew)

plt.ylabel('Recompensa final')
plt.xlabel('Cardinalidad de S')
plt.title("Simple Plot")
plt.legend()
plt.show()
```

Bibliografía

[1] Ross S. (1983). Introduction to Stochastic Dynamic Programming, Academic Press, New York.

[2] John Bather. (2000). Decision Theory: An Introduction to Dynamic Programming and Sequential Decisions, John Wiley & Sons, Inc., New York.

[3] García, Émilien. (2016). Estudio de variantes del problema de la Secretaria. Santiago, Chile: Universidad de Chile - Facultad de Ciencias Físicas y Matemáticas. http://repositorio.uchile.cl/handle/2250/137963

[4] P. R. Freeman. (1983). The Secretary Problem and Its Extensions: A Review, International Statistical Review,(51)2, 189-206.

[5] R. Bellman. (1957). A markovian decision process, Journal of Mathematics and Mechanics, 679–684.

[6] O. Hernández-Lerma. (1996). Discrete-time Markov control processes: basic optimality criteria, Springer Science & Business Media, New York.

[7] A. Jaśkiewicz y A. S. Nowak. (2011). Discounted dynamic programming with unbounded returns: application to economic models, Journal of Mathematical Analysis and Applications, 450-462.