Evaluación de las decisiones tomadas por un equipo durante un partido de fútbol

2024-11-19

Tabla de contenidos

Introducción **3**

1 Formulación **8**

2 Dinámica del modelo **11**

3 Descripción y Justificación de la recompensa **18**

4 Justificación de las acciones **19**

5 Simulación **20**
5.1 Club Deportido Guadalajara 20
5.2 Club América 24
5.3 Cruz Azul 28

References **33**

Introducción

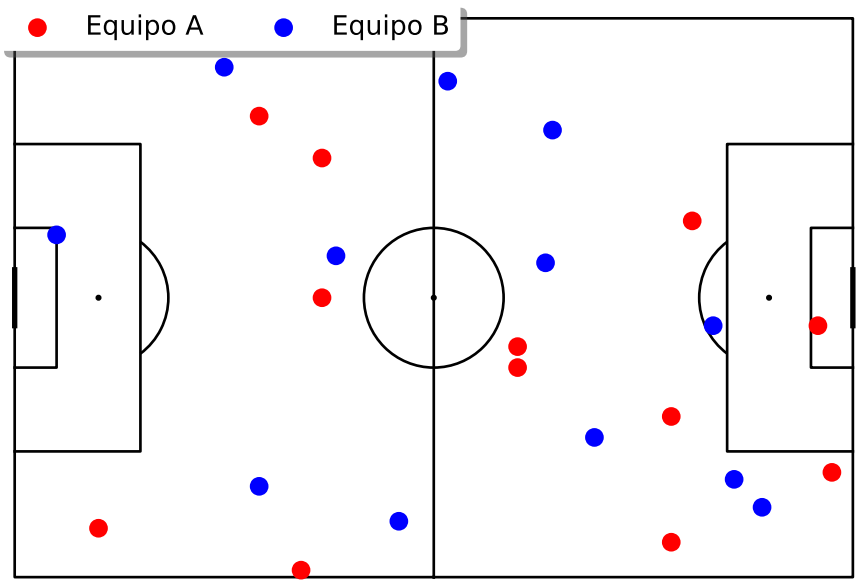
Este trabajo está basado en los trabajos: Van Roy, Yang, et al. (2021) y Van Roy, Robberechts, et al. (2021).

Para la construcción del Proceso de Decisión de Markov (MDP) planteado en este trabajo, primero debemos recordar algunas reglas durante un partido de fútbol para aquellos que no las conozcan y también hacer algunas puntualizaciones respecto a la naturaleza de un partido.

Además, se darán algunas hipótesis para simplificar el modelo sin pérder la esencia del juego.

Un partido de fútbol se compone de 2 equipos con 11 jugadores cada uno, un balón y un árbitro (o un grupo de árbitros). Los equipos se posicionan dentro de un terreno de juego que consta de 2 portería (una para cada equipo), el objetivo de cada equipo es anotar un gol (que el balón ingrese por completo dentro de la portería) en la portería del equipo contrario. Por cada gol un equipo recibe lo equivalente a 1 punto, al final del partido el equipo que anote más goles gana y en caso de anotar la misma cantidad se considera empate.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mplsoccer import Pitch, FontManager, add_image
import random
pitch = Pitch(line_color='black',linewidth=1)
fig, ax = pitch.draw()

pitch.scatter(random.randint(1,119),random.randint(1,79),color = 'red', ax=ax, label = 'Eq
pitch.scatter(random.randint(1,119),random.randint(1,79),color = 'blue', ax=ax, label = 'E
for i in range(0,11):
    pitch.scatter(random.randint(1,119),random.randint(1,79),color = 'red', ax=ax)
    pitch.scatter(random.randint(1,119),random.randint(1,79),color = 'blue', ax=ax)



ax.legend(shadow=True, loc = 'upper left', ncol=2, facecolor = 'white', edgecolor = 'None'
```



Para anotar un gol cada jugador puede tocar el balón con cualquier parte del cuerpo salvo el brazo, sin embargo, de los 11 jugadores de cada equipo hay un futbolista que puede tomar el balón con las manos dentro de las áreas marcadas alrededor de las porterías, a este jugador se le conoce como portero.

Un partido consta de 90 minutos divididos en 2 partes de 45 minutos.

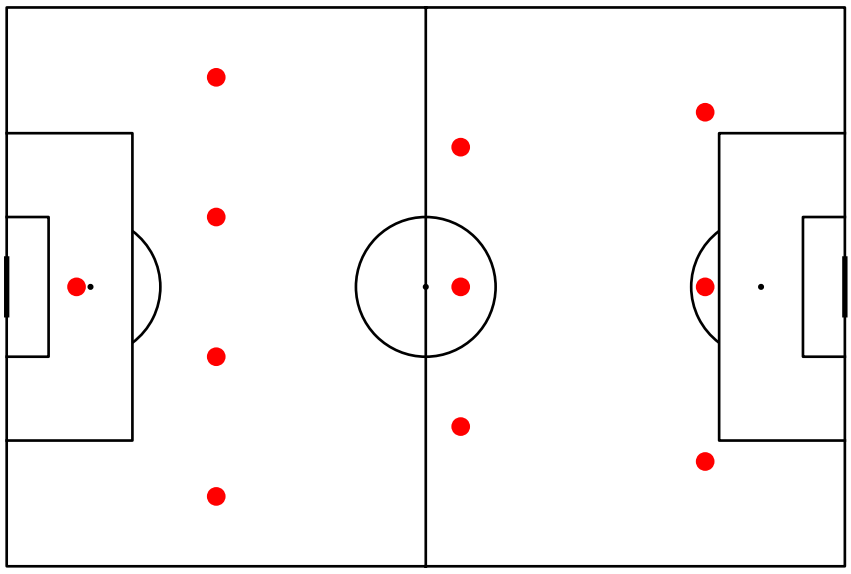
Explicadas las reglas principales vamos a familiarizarnos con el desarrollo de un partido.

Si bien no existen reglas escritas sobre cómo se deban acomodar los 11 jugadores en el campo, durante la historia siempre se han acomodado de la siguiente forma:

- El portero siempre se mantiene dentro del área y lo más cercano a la portería.
- Existe un primer grupo llamados defensas que se acomodan delante del portero y son los encargados que el equipo contrario no marque gol.
- Existe un segundo grupo llamados mediocampistas que se acomodan delante de los defensas y que su función es trasladar el balón de los defensas hacia la portería rival.
- Existe un tercer grupo llamados delanteros que son los encargados de recibir los balones de los mediocampistas y tratar de anotar gol.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mplsoccer import Pitch, FontManager, add_image
import random
pitch = Pitch(line_color='black',linewidth=1)
fig, ax = pitch.draw()
```

```
# Portero
pitch.scatter(10,40,color = 'red', ax=ax)

# Defensas
for i in range (0,4):
    a = 10 + i*20
    pitch.scatter(30,a,color = 'red', ax=ax)

# Mediocampistas
for i in range (0,3):
    a = 20 + i*20
    pitch.scatter(65,a,color = 'red', ax=ax)

# Delanteros
for i in range (0,3):
    a = 15 + i*25
    pitch.scatter(100,a,color = 'red', ax=ax)
```



El problema principal al que nos enfrentamos y por lo cual el fútbol es tan difícil de analizar es porque las descripciones anteriores son muy generales y muchas veces no se cumplen. Es decir, podemos tener a un defensa o un mediocampista metiendo goles sin necesidad de los delanteros, podemos tener mediocampistas defendiendo o se puede anotar un gol pasando directamente de los defensas o el portero hasta los delanteros. En cuestión de dinámica casi todo está permitido, esto no quiere decir que suceda, sin embargo, es muy complicado analizar de forma mecánica cada decisión de los futbolistas.

Aún así podemos enlistar ciertas situaciones que no son comunes que pasen y por tanto para nuestro modelo las vamos a ignorar:

- No se anotan goles desde la primera mitad de campo (desde la portería propia hasta el medio campo).
- Los porteros no anotan goles.
- La única forma de anotar gol es mediante una acción que llamamos tiros.
- Cada equipo tiene 11 jugadores.

También durante el desarrollo de un partido hay múltiples acciones que puede realizar un futbolista, a continuación, enlistamos algunas de las acciones:

- *pase* : Golpear el balón con el pie, cabeza o alguna otra parte del cuerpo con intención que el balón llegue a un jugador del mismo equipo.

https://www.youtube.com/watch?v=f-Bw-UaVV4U&ab_channel=SportsHD

- *tiro* : Golpear el balón con el pie, cabeza o alguna otra parte del cuerpo con intención que el balón ingrese en la portería contraria.

https://www.youtube.com/watch?v=si1RC4w-FAI&ab_channel=90Minutes

- *regate* : Mover el balón de forma que el jugador rival no me lo quite y me haga avanzar o terminar en una posición favorable.

https://www.youtube.com/watch?v=mathV4xFUX8&ab_channel=VSVHD

- *centro* : Golpear el balón con el pie, cabeza o alguna otra parte del cuerpo con intención que el balón llegue por el aire a un jugador del mismo equipo.

https://www.youtube.com/watch?v=-nI94rkcOwk&t=11s&ab_channel=rom7ooo

- *tackle* : Detener el avance de un jugador contrario.

https://www.youtube.com/watch?v=ZIX8b85FsQ8&ab_channel=7XFootball

Existen muchas más acciones que puede realizar un futbolista, sin embargo, las enlistadas anteriormente son las más comunes y muchas se desprenden de ellas.

Con todo lo anterior podemos dejar claro cuál es el propósito del proyecto.

Como mencionamos lo más importante durante un partido son los goles, hay que marcar goles y evitar que nos marquen gol. En este proyecto nos centramos en la parte ofensiva, es decir marcar goles. De esta forma nuestro propósito es el de encontrar la mejor forma de mover el balón por el campo que nos haga marcar un gol.

Para lograr esto se van a hacer algunas hipótesis:

- El campo se divide en 3 secciones iguales: Defensa, Mediocampo, Delantera.
- Solo se consideran 3 acciones: pase, tiro, regate.
- Los jugadores toman decisiones coherentes.

- No se consideran: fuera de juego, tiros libres, penales, saques de banda.
- Se considera al equipo completo y no a cada jugador individualmente.

Llamaremos jugada a una serie de acciones y puntualizar que:

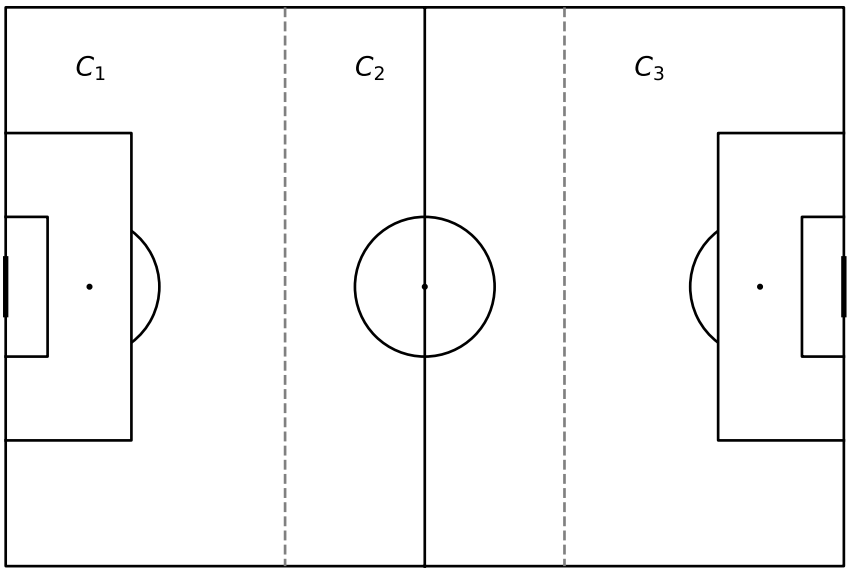
- Una jugada puede comenzar en cualquier parte del campo.
- Una jugada termina cuando realizamos un tiro o cuando perdemos el balón.

1 Formulación

El Proceso de Decisión de Markov se compone de los siguientes elementos:

- El conjunto de estados estará conformado por las siguientes tres divisiones del campo

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import to_rgba
from urllib.request import urlopen
from mplsoccer import Pitch, FontManager, add_image
from PIL import Image

pitch = Pitch(line_color='black',linewidth=1)
fig, ax = pitch.draw()
pitch.lines(40,0,40,80,color = 'gray', linewidth = 1, ax=ax, linestyle = '--')

pitch.lines(80,0,80,80,color = 'gray', linewidth = 1, ax=ax, linestyle = '--')

pitch.annotate(text='$C_1$',xytext=(10,10),xy=(84,45),ax=ax)
pitch.annotate(text='$C_2$',xytext=(50,10),xy=(84,45),ax=ax)
pitch.annotate(text='$C_3$',xytext=(90,10),xy=(84,45),ax=ax)
plt.show()
```



Donde C_1, C_2, C_3 representa que el balón se encuentre en alguna de las tres divisiones.

Además se agregan tres estados absorbentes:

- L_p = pérdida de posesión del balón.
- nG = realizar un tiro y que no termine en gol.
- G = realizar un tiro y que termine en gol.

De esta forma el conjunto de estados $\mathcal{S}$ queda como

$$\mathcal{S} = \{C_1, C_2, C_3, L_p, nG, G\}$$

- El conjunto de acciones admisibles se considerarán 3 acciones que serán

  - t = tiro
  - p = pase
  - r = regate

  De esta forma el conjunto de acciones queda como

  $$\mathcal{A} = \{t, p, r\}$$

- Para las de transiciones haremos uso de las probabilidades de transición definidas de la siguiente forma:

  $$P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$$

  Que se interpreta como la probabilidad de estar en un estado S_i realizar una acción a_k y terminar en un estado S_j. Notemos que se aceptan los casos cuando $i = j$ y más adelante se mostrará que algunas probabilidades serán 0.

- La función de recompensa $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, será

$$R(S_i, a_k, S_j) = \begin{cases} 1 & \text{si} \quad S_j = G \\ 0 & o.c. \end{cases}$$

2 Dinámica del modelo

En el contexto del fútbol llamamos *jugada* a una sucesión de acciones donde el balón se traslada desde un punto inicial donde el equipo A tiene el balón hasta un punto final que puede ser: perder el balón, tirar a puerta y no anotar gol o tirar y anotar gol.

Ejemplo: El balón comienza en el saque de meta del portero, el portero da un pase a un defensa que se encuentra en el primer tercio, que esté da un pase a un delantero que se encuentra en el tercer tercio y al intentar un regate pierde el balón.

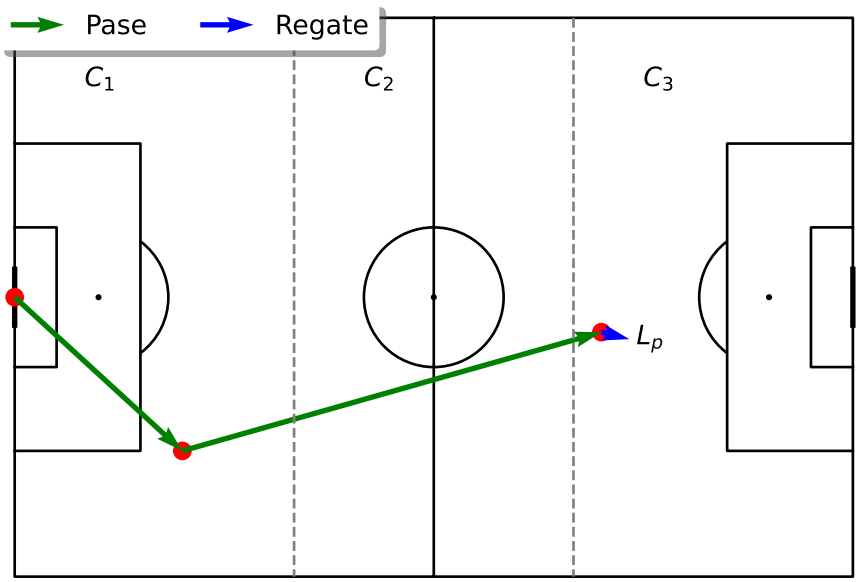
En nuestro contexto se verá como el hecho de iniciar la sucesión de acciones desde alguna sección C_i y terminar en alguno de los 3 estados absorbentes.

Ejemplo

$$C_1 \xrightarrow{p} C_1 \xrightarrow{p} C_3 \xrightarrow{r} L_p.$$

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mplsoccer import Pitch, FontManager, add_image

pitch = Pitch(line_color='black',linewidth=1)
fig, ax = pitch.draw()
pitch.lines(40,0,40,80,color = 'gray', linewidth = 1, ax=ax, linestyle = '--')

pitch.lines(80,0,80,80,color = 'gray', linewidth = 1, ax=ax, linestyle = '--')

pitch.annotate(text='$C_1$',xytext=(10,10),xy=(84,45),ax=ax)
pitch.annotate(text='$C_2$',xytext=(50,10),xy=(84,45),ax=ax)
pitch.annotate(text='$C_3$',xytext=(90,10),xy=(84,45),ax=ax)

pitch.scatter(0,40, color = 'red', ax=ax)
pitch.scatter(24,62,color = 'red',ax=ax)
pitch.scatter(84,45,color = 'red',ax=ax)

pitch.arrows(0,40,24,62,color = 'green', width = 2,label = 'Pase', ax=ax)
pitch.arrows(24,62,84,45,color = 'green', width = 2, ax=ax)
pitch.arrows(84,45,88,46,color = 'blue', width = 2,label = 'Regate', ax=ax)
pitch.annotate(text='$L_p$',xytext=(89,47),xy=(84,45),ax=ax)

ax.legend(shadow=True, loc = 'upper left', ncol=2, facecolor = 'white', edgecolor = 'None'
```

```
plt.show()
```



Para movernos de un estado S_i a un estado S_j mediante una acción a_k haremos uso de las probabilidades de transición, estas probabilidades las estimaremos utilizando datos extraídos de FBREF para 4 clubes: Chivas, América y Cruz Azul de la temporada 2023-2024

Primero vamos a interpretar las probabilidades de transición con la finalidad de descartar aquellas transciones que no serán posibles con nuestro modelo y con la naturaleza de un partido.

- Fijamos el estado C_1.
  - Consideramos la acción p:

    $P(C_1, p, C_1)$ = La probabilidad de estar en la zona C_1 dar un *pase* y terminar en la zona C_1.

    $P(C_1, p, C_2)$ = La probabilidad de estar en la zona C_1 dar un *pase* y terminar en la zona C_2.

    $P(C_1, p, C_3)$ = La probabilidad de estar en la zona C_1 dar un *pase* y terminar en la zona C_3.

    $P(C_1, p, L_p)$ = La probabilidad de estar en la zona C_1 dar un *pase* y perder el balón.

    $P(C_1, p, nG)$ = La probabilidad de estar en la zona C_1 dar un *pase* y no anotar gol.

    $P(C_1, p, G)$ = La probabilidad de estar en la zona C_1 dar un *pase* y anotar gol.

De esta lista de probabilidades para la acción p notemos que $P(C_1, p, nG) = P(C_1, p, G) = 0$ esto pues la única acción admisible para terminar en los estados $\{nG, G\}$ es t.

- Consideramos la acción r:

  $P(C_1, r, C_1)$ = La probabilidad de estar en la zona C_1 hacer un *regate* y terminar en la zona C_1.

  $P(C_1, r, C_2)$ = La probabilidad de estar en la zona C_1 hacer un *regate* y terminar en la zona C_2.

  $P(C_1, r, C_3)$ = La probabilidad de estar en la zona C_1 hacer un *regate* y terminar en la zona C_3.

  $P(C_1, r, L_p)$ = La probabilidad de estar en la zona C_1 hacer un *regate* y perder el balón.

  $P(C_1, r, nG)$ = La probabilidad de estar en la zona C_1 hacer un *regate* y no anotar gol.

  $P(C_1, r, G)$ = La probabilidad de estar en la zona C_1 hacer un *regate* y anotar gol.

  De esta lista de probabilidades para la acción r notemos que $P(C_1, r, nG) = P(C_1, r, nG) = 0$ esto pues la única acción admisible para terminar en los estados nG, G es t.

  Además $P(C_1, r, C_3) = 0$, pues al realizar un *regate* solo tenemos dos opciones:

  nos mantenemos en la zona C_1 o avanzamos a la zona siguiente C_2.

- Consideramos la acción t:

  En este caso tendremos que $P(C_1, t, C_1) = P(C_1, t, C_2) = P(C_1, t, C_3) = P(C_1, t, L_p) = 0$ pues después de realizar un *tiro* solo tendremos dos estados posibles $\{nG, G\}$.

  $P(C_1, t, nG)$ = La probabilidad de estar en la zona C_1 hacer un *tiro* y no anotar gol.

  $P(C_1, t, G)$ = La probabilidad de estar en la zona C_1 hacer un *tiro* y anotar gol.

  Sin embargo, como suponemos que las acciones que toman las futbolistas son coherentes, no tiene sentido realizar un *tiro* desde la zona C_1, pues la distancia hacia la porteria contaría es muy lejana y la probabilidad de anotar un gol es prácticamente nula. Por tanto

$$P(C_1, t, S) = 0, \quad \forall S \in \mathcal{S}.$$

- Fijamos el estado C_2.

– Consideramos la acción p:

$P(C_2, p, C_1)$ = La probabilidad de estar en la zona C_2 dar un *pase* y terminar en la zona C_1.

$P(C_2, p, C_2)$ = La probabilidad de estar en la zona C_2 dar un *pase* y terminar en la zona C_2.

$P(C_2, p, C_3)$ = La probabilidad de estar en la zona C_2 dar un *pase* y terminar en la zona C_3.

$P(C_2, p, L_p)$ = La probabilidad de estar en la zona C_2 dar un *pase* y perder el balón.

$P(C_2, p, nG)$ = La probabilidad de estar en la zona C_2 dar un *pase* y no tirar a gol.

$P(C_2, p, G)$ = La probabilidad de estar en la zona C_2 dar un *pase* y terminar en gol.

De esta lista de probabilidades para la acción p notemos que $P(C_2, p, nG) = P(C_2, p, nG) = 0$ esto pues la única acción admisible para terminar en los estados $\{nG, G\}$ es t.

– Consideramos la acción r:

$P(C_2, r, C_1)$ = La probabilidad de estar en la zona C_2 hacer un *regate* y terminar en la zona C_1.

$P(C_2, r, C_2)$ = La probabilidad de estar en la zona C_2 hacer un *regate* y terminar en la zona C_2.

$P(C_2, r, C_3)$ = La probabilidad de estar en la zona C_2 hacer un *regate* y terminar en la zona C_3.

$P(C_2, r, L_p)$ = La probabilidad de estar en la zona C_2 hacer un *regate* y perder el balón.

$P(C_2, r, nG)$ = La probabilidad de estar en la zona C_2 hacer un *regate* y no anotar gol.

$P(C_2, r, G)$ = La probabilidad de estar en la zona C_2 hacer un *regate* y anotar gol.

De esta lista de probabilidades para la acción r notemos que $P(C_2, r, nG) = P(C_2, r, G) = 0$ esto pues la única acción admisible para terminar en los estados $\{nG, G\}$ es t.

– Consideramos la acción t:

En este caso tendremos que $P(C_2, t, C_1) = P(C_2, t, C_2) = P(C_2, t, C_3) = P(C_2, t, L_p) = 0$ pues después de realizar un *tiro* solo tendremos dos estados posibles $\{nG, G\}$.

$P(C_2, t, nG)$ = La probabilidad de estar en la zona C_2 hacer un *tiro* y no anotar gol.

$P(C_2, t, G)$ = La probabilidad de estar en la zona C_2 hacer un *tiro* y anotar gol.

- Fijamos el estado C_3
  - Consideramos la acción p:

    $P(C_3, p, C_1)$ = La probabilidad de estar en la zona C_3 dar un *pase* y terminar en la zona C_1.

    $P(C_3, p, C_2)$ = La probabilidad de estar en la zona C_3 dar un *pase* y terminar en la zona C_2.

    $P(C_3, p, C_3)$ = La probabilidad de estar en la zona C_3 dar un *pase* y terminar en la zona C_3.

    $P(C_3, p, L_p)$ = La probabilidad de estar en la zona C_3 dar un *pase* y perder el balón.

    $P(C_3, p, nG)$ = La probabilidad de estar en la zona C_3 dar un *pase* y no tirar a gol.

    $P(C_3, p, G)$ = La probabilidad de estar en la zona C_3 dar un *pase* y terminar en gol.

    De esta lista de probabilidades para la acción p notemos que $P(C_3, p, nG) = P(C_2, p, G) = 0$ esto pues la única acción admisible para terminar en los estados $\{nG, G\}$ es t.

  - Consideramos la acción r:

    $P(C_3, r, C_1)$ = La probabilidad de estar en la zona C_3 hacer un *regate* y terminar en la zona C_1.

    $P(C_3, r, C_2)$ = La probabilidad de estar en la zona C_3 hacer un *regate* y terminar en la zona C_2.

    $P(C_3, r, C_3)$ = La probabilidad de estar en la zona C_3 hacer un *regate* y terminar en la zona C_3.

    $P(C_3, r, L_p)$ = La probabilidad de estar en la zona C_3 hacer un *regate* y perder el balón.

    $P(C_3, r, nG)$ = La probabilidad de estar en la zona C_3 hacer un *regate* y no anotar gol.

    $P(C_3, r, G)$ = La probabilidad de estar en la zona C_3 hacer un *regate* y anotar gol.

    De esta lista de probabilidades para la acción r notemos que $P(C_3, r, nG) = P(C_3, r, G) = 0$ esto pues la única acción admisible para terminar en los estados $\{nG, G\}$ es t.

Además $P(C_3, r, C_1) = 0$, pues al realizar un *regate* solo tenemos dos opciones o nos mantenemos en la zona C_3 o retrocedemos a la zona anterior C_2.

– Consideramos la acción t:

En este caso tendremos que $P(C_3, t, C_1) = P(C_3, t, C_2) = P(C_3, t, C_3) = P(C_3, t, L_p) = 0$ pues después de realizar un *tiro* solo tendremos dos estados posibles $\{nG, G\}$.

$P(C_3, t, nG) =$ La probabilidad de estar en la zona C_3 hacer un *tiro* y no anotar gol.

$P(C_3, t, G) =$ La probabilidad de estar en la zona C_3 hacer un *tiro* y anotar gol.

- Por último como $\{L_p, nG, G\}$ son estados absorbentes, entonces $\forall a \in \mathcal{A}$.

$$P(L_p, a, S) = \begin{cases} 1 & \text{si} \quad S = L_p \\ 0 & o.c. \end{cases}$$

$$P(nG, a, S) = \begin{cases} 1 & \text{si} \quad S = nG \\ 0 & o.c. \end{cases}$$

$$P(G, a, S) = \begin{cases} 1 & \text{si} \quad S = G \\ 0 & o.c. \end{cases}$$

Para facilitar las estimaciones se renombrarán cada probabiidad con los parámetros que se muestran en las siguientes tablas

Tabla 2.1: Parámetros para las probabilidades de C_1

Probabilidades	Parámetros
$P(C_1, p, C_1)$	α_{1p}
$P(C_1, p, C_2)$	α_{2p}
$P(C_1, p, C_3)$	α_{3p}
$P(C_1, p, L_p)$	α_{4p}
$P(C_1, r, C_1)$	α_{1r}
$P(C_1, r, C_2)$	α_{2r}
$P(C_1, r, L_p)$	α_{3r}

Tabla 2.2: Parámetros para las probabilidades de C_2

Probabilidades	Parámetros
$P(C_2, p, C_1)$	β_{1p}
$P(C_2, p, C_2)$	β_{2p}
$P(C_2, p, C_3)$	β_{3p}

Probabilidades	Parámetros
$P(C_2, p, L_p)$	β_{4p}
$P(C_2, r, C_1)$	β_{1r}
$P(C_2, r, C_2)$	β_{2r}
$P(C_2, r, C_3)$	β_{3r}
$P(C_2, r, L_p)$	β_{4r}
$P(C_2, t, nG)$	β_{1t}
$P(C_2, t, G)$	β_{1t}

Tabla 2.3: Parámetros para las probabilidades de C_3

Probabilidades	Parámetros
$P(C_3, p, C_1)$	γ_{1p}
$P(C_3, p, C_2)$	γ_{2p}
$P(C_3, p, C_3)$	γ_{3p}
$P(C_3, p, L_p)$	γ_{4p}
$P(C_3, r, C_2)$	γ_{1r}
$P(C_3, r, C_3)$	γ_{2r}
$P(C_3, r, L_p)$	γ_{3r}
$P(C_3, t, nG)$	γ_{1t}
$P(C_3, t, G)$	γ_{2t}

3 Descripción y Justificación de la recompensa

En un partido de fútbol gana el equipo que anota más goles, en caso de anotar los mismos goles se considera empate y no existe desempate de ningún tipo.[1] Por lo que la recompensa será la de anotar un gol $R = 1$, pues es lo único que puede hacer que un equipo gane un partido. No existe penalización porque los goles válidos anotados no pueden ser descontandos.

[1] No se consideran los partidos de eliminación directa donde existe el desempate por penales.

4 Justificación de las acciones

Las acciones que puede realizar un equipo durante un partido son limitadas y se pueden enlistar. Sin embargo para nuestro modelo vamos a seleccionar las 3 más importantes que son el *pase, tiro* y *regate*.

- *tiro*: Es la acción que nos permite anotar goles.
- *pase*: Ayuda a un equipo a mover el balón por el campo sin necesidad de desplazarse o dejar rivales atrás.
- *regate*: Permite que podamos trasladar el balón de un lugar a otro y dejar a rivales atrás.

5 Simulación

Para completar este trabajo mostramos un código que nos ayuda a encontrar una politica óptima. Esto lo haremos estimando los parametros definidos en la sección Capítulo 2 para los 4 equipos (Chivas, América, Cruz Azul).

5.1 Club Deportido Guadalajara

Según datos extraídos de FBREF obtenemos los siguientes parámetros:

```
# Parámetros de transición y recompensas
# C_1
alpha_1p = 0.43
alpha_2p = 0.23
alpha_3p = 0.12
alpha_4p = 0.22
alpha_1r = 0.28
alpha_2r = 0.20
alpha_3r = 0.52

# C_2
beta_1p = 0.34
beta_2p = 0.21
beta_3p = 0.14
beta_4p = 0.31
beta_1r = 0.20
beta_2r = 0.12
beta_3r = 0.07
beta_4r = 0.61
beta_1t = 0.01
beta_2t = 0.99

# C_3
gamma_1p = 0.05
gamma_2p = 0.47
gamma_3p = 0.18
gamma_4p = 0.30
gamma_1r = 0.14
```

```
gamma_2r = 0.08
gamma_3r = 0.78
gamma_1t = 0.10
gamma_2t = 0.90
```

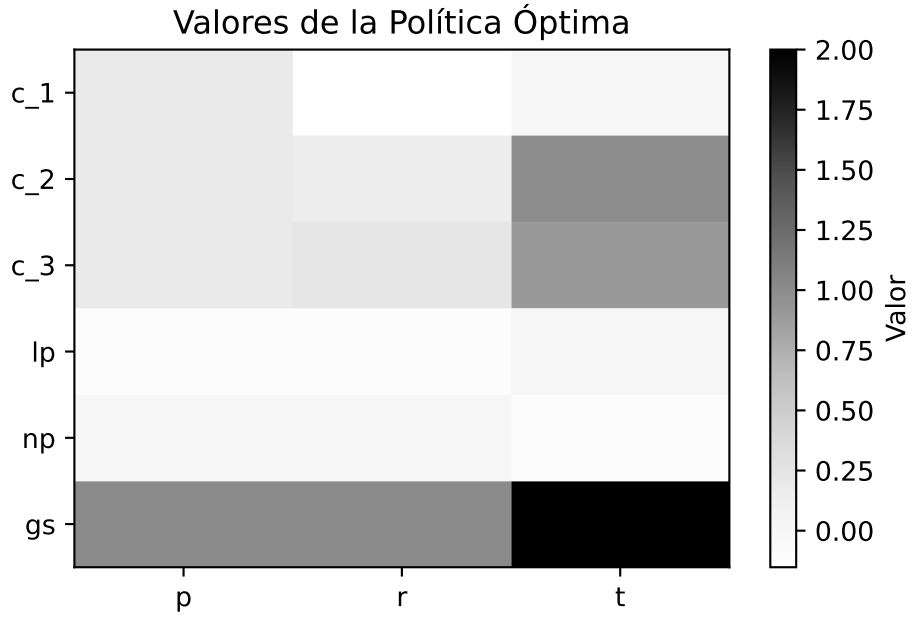
Con esto podemos obtener la política óptima. Además por motivos de simulación agregamos ciertos costo y recompensas para cada estado dependiendo de la acción, esto se me modificará

```
import numpy as np
import matplotlib.pyplot as plt
# Parámetros de transición y recompensas
# C_1
alpha_1p = 0.43
alpha_2p = 0.23
alpha_3p = 0.12
alpha_4p = 0.22
alpha_1r = 0.28
alpha_2r = 0.20
alpha_3r = 0.52

# C_2
beta_1p = 0.34
beta_2p = 0.21
beta_3p = 0.14
beta_4p = 0.31
beta_1r = 0.20
beta_2r = 0.12
beta_3r = 0.07
beta_4r = 0.61
beta_1t = 0.01
beta_2t = 0.99

# C_3
gamma_1p = 0.05
gamma_2p = 0.47
gamma_3p = 0.18
gamma_4p = 0.30
gamma_1r = 0.14
gamma_2r = 0.08
gamma_3r = 0.78
gamma_1t = 0.10
gamma_2t = 0.90

# Estados y acciones
```

```
states = ['c_1','c_2','c_3','lp','np','gs']
actions = ['p','r','t']
rewards = {
    'c_1':{'p':0.1,'r':-0.2,'t':0},
    'c_2':{'p':0.1,'r':0.1,'t':0},
    'c_3':{'p':0.1,'r':0.2,'t':0},
    'lp':{'p':-0.1,'r':-0.1,'t':0},
    'np':{'p':0,'r':0,'t':-0.1},
    'gs':{'p':0,'r':0,'t':1}}
transition = {
    'c_1':{'p':{'c_1':alpha_1p,'c_2':alpha_2p,'c_3':alpha_3p,'lp':alpha_4p,'np':0,'gs':0},
           'r':{'c_1':alpha_1r,'c_2':alpha_2r,'c_3':0,'lp':alpha_3r,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':0}},

    'c_2':{'p':{'c_1':beta_1p,'c_2':beta_2p,'c_3':beta_3p,'lp':beta_4p,'np':0,'gs':0},
           'r':{'c_1':beta_1r,'c_2':beta_2r,'c_3':beta_3r,'lp':beta_4r,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':beta_1t,'gs':beta_2t}},

    'c_3':{'p':{'c_1':gamma_1p,'c_2':gamma_2p,'c_3':gamma_3p,'lp':gamma_4p,'np':0,'gs':0},
           'r':{'c_1':0,'c_2':gamma_1r,'c_3':gamma_2r,'lp':gamma_3r,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':gamma_1t,'gs':gamma_2t}},

    'lp':{'p':{'c_1':0,'c_2':0,'c_3':0,'lp':1,'np':0,'gs':0},
           'r':{'c_1':0,'c_2':0,'c_3':0,'lp':1,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':1,'np':0,'gs':0}},

    'np':{'p':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':1,'gs':0},
           'r':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':1,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':1,'gs':0}},

    'gs':{'p':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':1},
           'r':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':1},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':1}}}

# Función para Value Iteration
def value_iteration():
    V = {s: 0 for s in states}
    cont = 0
    while (True and cont<10000):
        new_V = {}
        for s in states:
            values = []
            for a in actions:
                value = rewards[s][a]
                for s2 in states:
```

```
                    value += transition[s][a][s2] * V[s2]
                    #print(value)
                values.append(value)
            #print(values)
            new_V[s]= max(values)
            #print(f'nuevo vector {new_V}')

        if all(abs(V[s]-new_V[s])<0.0001 for s in states):
            return new_V
        cont = cont + 1
        V = new_V
        return V
V = value_iteration()

# Política óptima
policy = {}
for s in states:
    values = []
    for a in actions:
        value = rewards[s][a]
        for s2 in states:
            value+= transition[s][a][s2] * V[s2]
        values.append(value)
    policy[s] = actions[np.argmax(values)]
print('Política Óptima')
print(policy)

# Visualización de los valores de política
policy_values = np.zeros((len(states),len(actions)))
for i, s in enumerate(states):
    for j, a in enumerate(actions):
        policy_values[i, j] = rewards[s][a] + sum(transition[s][a][s2] * V[s2] for s2 in s
plt.imshow(policy_values, cmap="Greys", aspect="auto")
plt.xticks(ticks=range(len(actions)), labels=actions)
plt.yticks(ticks=range(len(states)), labels=states)
plt.colorbar(label="Valor")
plt.title("Valores de la Política Óptima")
plt.show()
```

```
Política Óptima
{'c_1': 'p', 'c_2': 't', 'c_3': 't', 'lp': 't', 'np': 'p', 'gs': 't'}
```



5.2 Club América

Según datos extraídos de FBREF obtenemos los siguientes parámetros:

```
# Parámetros de transición y recompensas
# C_1
alpha_1p = 0.59
alpha_2p = 0.27
alpha_3p = 0.02
alpha_4p = 0.12
alpha_1r = 0.54
alpha_2r = 0.39
alpha_3r = 0.07

# C_2
beta_1p = 0.31
beta_2p = 0.45
beta_3p = 0.12
beta_4p = 0.12
beta_1r = 0.29
beta_2r = 0.24
beta_3r = 0.16
beta_4r = 0.31
beta_1t = 0.004
beta_2t = 0.996
```

```
# C_3
gamma_1p = 0.09
gamma_2p = 0.58
gamma_3p = 0.10
gamma_4p = 0.23
gamma_1r = 0.14
gamma_2r = 0.10
gamma_3r = 0.76
gamma_1t = 0.17
gamma_2t = 0.83
```

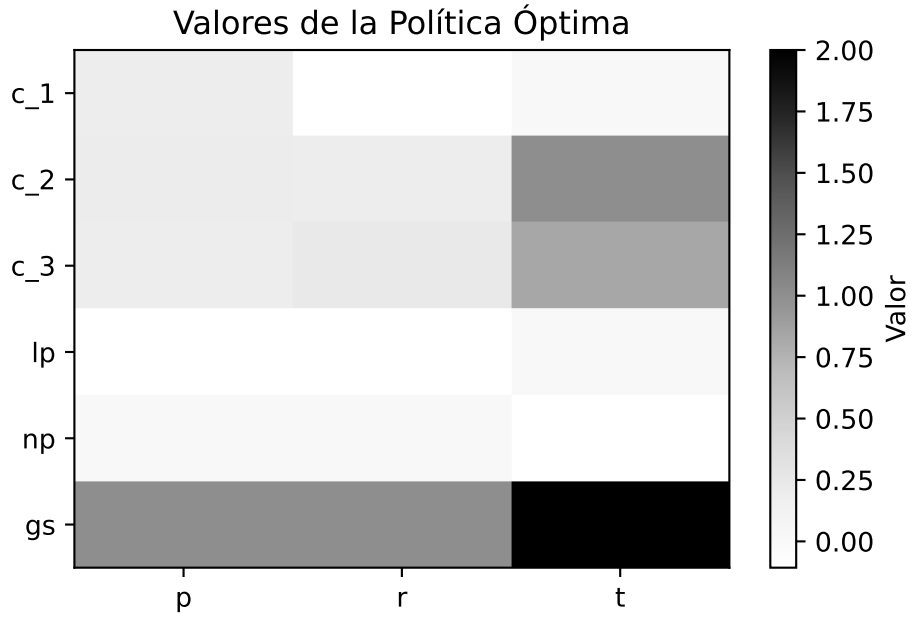
Con esto podemos obtener la política óptima. Además por motivos de simulación agregamos ciertos costo y recompensas para cada estado dependiendo de la acción.

```
import numpy as np
import matplotlib.pyplot as plt
# Parámetros de transición y recompensas
# C_1
alpha_1p = 0.59
alpha_2p = 0.27
alpha_3p = 0.02
alpha_4p = 0.12
alpha_1r = 0.54
alpha_2r = 0.39
alpha_3r = 0.07

# C_2
beta_1p = 0.31
beta_2p = 0.45
beta_3p = 0.12
beta_4p = 0.12
beta_1r = 0.29
beta_2r = 0.24
beta_3r = 0.16
beta_4r = 0.31
beta_1t = 0.004
beta_2t = 0.996

# C_3
gamma_1p = 0.09
gamma_2p = 0.58
gamma_3p = 0.10
gamma_4p = 0.23
gamma_1r = 0.14
```

```
gamma_2r = 0.10
gamma_3r = 0.76
gamma_1t = 0.17
gamma_2t = 0.83

# Estados y acciones
states = ['c_1','c_2','c_3','lp','np','gs']
actions = ['p','r','t']
rewards = {
    'c_1':{'p':0.1,'r':-0.2,'t':0},
    'c_2':{'p':0.1,'r':0.1,'t':0},
    'c_3':{'p':0.1,'r':0.2,'t':0},
    'lp':{'p':-0.1,'r':-0.1,'t':0},
    'np':{'p':0,'r':0,'t':-0.1},
    'gs':{'p':0,'r':0,'t':1}}
transition = {
    'c_1':{'p':{'c_1':alpha_1p,'c_2':alpha_2p,'c_3':alpha_3p,'lp':alpha_4p,'np':0,'gs':0},
           'r':{'c_1':alpha_1r,'c_2':alpha_2r,'c_3':0,'lp':alpha_3r,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':0}},

    'c_2':{'p':{'c_1':beta_1p,'c_2':beta_2p,'c_3':beta_3p,'lp':beta_4p,'np':0,'gs':0},
           'r':{'c_1':beta_1r,'c_2':beta_2r,'c_3':beta_3r,'lp':beta_4r,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':beta_1t,'gs':beta_2t}},

    'c_3':{'p':{'c_1':gamma_1p,'c_2':gamma_2p,'c_3':gamma_3p,'lp':gamma_4p,'np':0,'gs':0},
           'r':{'c_1':0,'c_2':gamma_1r,'c_3':gamma_2r,'lp':gamma_3r,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':gamma_1t,'gs':gamma_2t}},

    'lp':{'p':{'c_1':0,'c_2':0,'c_3':0,'lp':1,'np':0,'gs':0},
           'r':{'c_1':0,'c_2':0,'c_3':0,'lp':1,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':1,'np':0,'gs':0}},

    'np':{'p':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':1,'gs':0},
           'r':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':1,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':1,'gs':0}},

    'gs':{'p':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':1},
           'r':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':1},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':1}}}

# Función para Value Iteration
def value_iteration():
    V = {s: 0 for s in states}
    cont = 0
    while (True and cont<10000):
```

```
        new_V = {}
        for s in states:
            values = []
            for a in actions:
                value = rewards[s][a]
                for s2 in states:
                    value += transition[s][a][s2] * V[s2]
                    #print(value)
                values.append(value)
            #print(values)
            new_V[s]= max(values)
            #print(f'nuevo vector {new_V}')

        if all(abs(V[s]-new_V[s])<0.0001 for s in states):
            return new_V
        cont = cont + 1
        V = new_V
        return V
V = value_iteration()

# Política óptima
policy = {}
for s in states:
    values = []
    for a in actions:
        value = rewards[s][a]
        for s2 in states:
            value+= transition[s][a][s2] * V[s2]
        values.append(value)
    policy[s] = actions[np.argmax(values)]
print('Política Óptima')
print(policy)

# Visualización de los valores de política
policy_values = np.zeros((len(states),len(actions)))
for i, s in enumerate(states):
    for j, a in enumerate(actions):
        policy_values[i, j] = rewards[s][a] + sum(transition[s][a][s2] * V[s2] for s2 in s
plt.imshow(policy_values, cmap="Greys", aspect="auto")
plt.xticks(ticks=range(len(actions)), labels=actions)
plt.yticks(ticks=range(len(states)), labels=states)
plt.colorbar(label="Valor")
plt.title("Valores de la Política Óptima")
plt.show()
```

```
Política Óptima
{'c_1': 'p', 'c_2': 't', 'c_3': 't', 'lp': 't', 'np': 'p', 'gs': 't'}
```



5.3 Cruz Azul

Según datos extraídos de FBREF obtenemos los siguientes parámetros:

```
# Parámetros de transición y recompensas
# C_1
alpha_1p = 0.63
alpha_2p = 0.28
alpha_3p = 0.01
alpha_4p = 0.08
alpha_1r = 0.52
alpha_2r = 0.29
alpha_3r = 0.19

# C_2
beta_1p = 0.41
beta_2p = 0.33
beta_3p = 0.12
beta_4p = 0.14
beta_1r = 0.19
beta_2r = 0.28
beta_3r = 0.11
```

```
beta_4r = 0.52
beta_1t = 0.009
beta_2t = 0.991

# C_3
gamma_1p = 0.10
gamma_2p = 0.37
gamma_3p = 0.19
gamma_4p = 0.34
gamma_1r = 0.21
gamma_2r = 0.09
gamma_3r = 0.70
gamma_1t = 0.08
gamma_2t = 0.92
```

Con esto podemos obtener la política óptima. Además por motivos de simulación agregamos ciertos costo y recompensas para cada estado dependiendo de la acción.

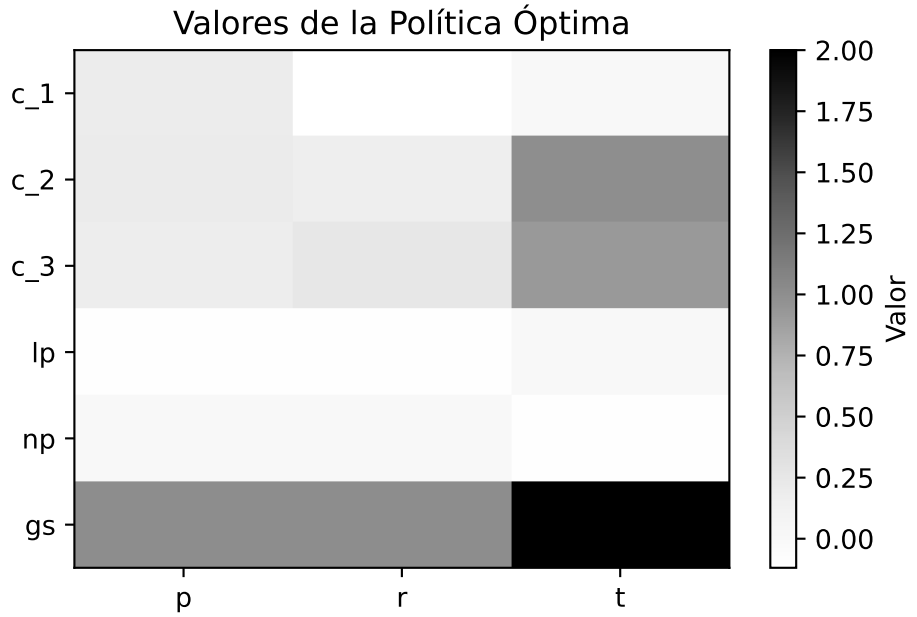
```
import numpy as np
import matplotlib.pyplot as plt

# Parámetros de transición y recompensas
# C_1
alpha_1p = 0.63
alpha_2p = 0.28
alpha_3p = 0.01
alpha_4p = 0.08
alpha_1r = 0.52
alpha_2r = 0.29
alpha_3r = 0.19

# C_2
beta_1p = 0.41
beta_2p = 0.33
beta_3p = 0.12
beta_4p = 0.14
beta_1r = 0.19
beta_2r = 0.28
beta_3r = 0.11
beta_4r = 0.52
beta_1t = 0.009
beta_2t = 0.991

# C_3
gamma_1p = 0.10
```

```
gamma_2p = 0.37
gamma_3p = 0.19
gamma_4p = 0.34
gamma_1r = 0.21
gamma_2r = 0.09
gamma_3r = 0.70
gamma_1t = 0.08
gamma_2t = 0.92

# Estados y acciones
states = ['c_1','c_2','c_3','lp','np','gs']
actions = ['p','r','t']
rewards = {
    'c_1':{'p':0.1,'r':-0.2,'t':0},
    'c_2':{'p':0.1,'r':0.1,'t':0},
    'c_3':{'p':0.1,'r':0.2,'t':0},
    'lp':{'p':-0.1,'r':-0.1,'t':0},
    'np':{'p':0,'r':0,'t':-0.1},
    'gs':{'p':0,'r':0,'t':1}}
transition = {
    'c_1':{'p':{'c_1':alpha_1p,'c_2':alpha_2p,'c_3':alpha_3p,'lp':alpha_4p,'np':0,'gs':0},
           'r':{'c_1':alpha_1r,'c_2':alpha_2r,'c_3':0,'lp':alpha_3r,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':0}},

    'c_2':{'p':{'c_1':beta_1p,'c_2':beta_2p,'c_3':beta_3p,'lp':beta_4p,'np':0,'gs':0},
           'r':{'c_1':beta_1r,'c_2':beta_2r,'c_3':beta_3r,'lp':beta_4r,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':beta_1t,'gs':beta_2t}},

    'c_3':{'p':{'c_1':gamma_1p,'c_2':gamma_2p,'c_3':gamma_3p,'lp':gamma_4p,'np':0,'gs':0},
           'r':{'c_1':0,'c_2':gamma_1r,'c_3':gamma_2r,'lp':gamma_3r,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':gamma_1t,'gs':gamma_2t}},

    'lp':{'p':{'c_1':0,'c_2':0,'c_3':0,'lp':1,'np':0,'gs':0},
           'r':{'c_1':0,'c_2':0,'c_3':0,'lp':1,'np':0,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':1,'np':0,'gs':0}},

    'np':{'p':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':1,'gs':0},
           'r':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':1,'gs':0},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':1,'gs':0}},

    'gs':{'p':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':1},
           'r':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':1},
           't':{'c_1':0,'c_2':0,'c_3':0,'lp':0,'np':0,'gs':1}}}

# Función para Value Iteration
```

```
def value_iteration():
    V = {s: 0 for s in states}
    cont = 0
    while (True and cont<10000):
        new_V = {}
        for s in states:
            values = []
            for a in actions:
                value = rewards[s][a]
                for s2 in states:
                    value += transition[s][a][s2] * V[s2]
                    #print(value)
                values.append(value)
            #print(values)
            new_V[s]= max(values)
            #print(f'nuevo vector {new_V}')

        if all(abs(V[s]-new_V[s])<0.0001 for s in states):
            return new_V
        cont = cont + 1
        V = new_V
        return V
V = value_iteration()

# Política óptima
policy = {}
for s in states:
    values = []
    for a in actions:
        value = rewards[s][a]
        for s2 in states:
            value+= transition[s][a][s2] * V[s2]
        values.append(value)
    policy[s] = actions[np.argmax(values)]
print('Política Óptima')
print(policy)

# Visualización de los valores de política
policy_values = np.zeros((len(states),len(actions)))
for i, s in enumerate(states):
    for j, a in enumerate(actions):
        policy_values[i, j] = rewards[s][a] + sum(transition[s][a][s2] * V[s2] for s2 in s
plt.imshow(policy_values, cmap="Greys", aspect="auto")
plt.xticks(ticks=range(len(actions)), labels=actions)
plt.yticks(ticks=range(len(states)), labels=states)
```

```
plt.colorbar(label="Valor")
plt.title("Valores de la Política Óptima")
plt.show()
```

```
Política Óptima
{'c_1': 'p', 'c_2': 't', 'c_3': 't', 'lp': 't', 'np': 'p', 'gs': 't'}
```



References

Van Roy, Maaike, Pieter Robberechts, Wen-Chi Yang, Luc De Raedt, y Jesse Davis. 2021. "Leaving goals on the pitch: Evaluating decision making in soccer". *arXiv preprint arXiv:2104.03252*.

Van Roy, Maaike, Wen-Chi Yang, Luc De Raedt, y Jesse Davis. 2021. "Analyzing learned markov decision processes using model checking for providing tactical advice in professional soccer". En *AI for Sports Analytics (AISA) Workshop at IJCAI 2021*.