1. Recordatorio: Juego Bimatriz y Equilibrio de Nash

1.1. Juego Bimatriz

Un **juego de dos jugadores** (Jugador 1 y Jugador 2) en **forma normal** puede describirse con dos matrices (o "bimatriz") de pagos:

- U^1 (de dimensión $m \times n$) para Jugador 1.
- U^2 (también $m \times n$) para Jugador 2.

Supongamos:

- Jugador 1 (fila) tiene m estrategias puras (filas).
- Jugador 2 (columna) tiene n estrategias puras (columnas).

Si Jugador 1 juega la fila i y Jugador 2 la columna j, entonces:

- Jugador 1 obtiene una utilidad (pago) U^1_{ij}.
- Jugador 2 obtiene una utilidad (pago) U^2_{ij}.

A diferencia de los **juegos de suma cero**, aquí no se asume $U^1_{ij} + U^2_{ij} = 0$; pueden ser de "suma no constante".

1.2. Estrategias Mixtas y Equilibrio de Nash

- Una **estrategia mixta** de Jugador 1 es un vector $\mathbf{x} = (x_1,\dots,x_m)$ con $x_i \ge 0$ y $\sum_i x_i = 1$. (Lo mismo para Jugador 2: $\mathbf{y} = (y_1,\dots,y_n)$.)
- La **utilidad esperada** para Jugador 1 si juega $\mathbf{x}$ contra $\mathbf{y}$ es $\mathbf{x}^\top U^1 \mathbf{y}$. Para Jugador 2, $\mathbf{x}^\top U^2 \mathbf{y}$.

$\textbf{Definición (Equilibrio de Nash).}\quad (\mathbf{x}^*, \mathbf{y}^*) \ \text{es un Equilibrio de Nash (EN) si:}$

1. Dada $\mathbf{y}^*$, $\mathbf{x}^*$ es **mejor respuesta** para Jugador 1. Es decir, para cualquier otra $\mathbf{x}$,
   $\mathbf{x}^T U^1 \mathbf{y}^* \ \le\ (\mathbf{x}^*)^T U^1 \mathbf{y}^*.$

2. Dada $\mathbf{x}^*$, $\mathbf{y}^*$ es **mejor respuesta** para Jugador 2. Es decir, para cualquier otra $\mathbf{y}$,
   $\mathbf{x}^{*T} U^2 \mathbf{y} \ \le\ \mathbf{x}^{*T} U^2 \mathbf{y}^*.$

En palabras simples: **cada jugador no puede mejorar su utilidad esperada cambiando unilateralmente** su estrategia mixta, asumiendo que el otro permanece con la suya.

---

2. El Algoritmo Lemke–Howson (Intuición)

Para **juegos bimatriz de 2 jugadores**, uno de los métodos más famosos para encontrar (al menos un) equilibrio de Nash es el **algoritmo de Lemke–Howson** (LH). La complejidad en el peor caso puede ser exponencial, pero se trata de un método de **pivoting** (similar a símplex en PL) que siempre encuentra **algún** equilibrio.

Idea geométrica:

- Se construye el **poliedro** de mejores respuestas para cada jugador.
- Cada estrategia pura de un jugador "entra en soporte" de la mezcla si es jugada con probabilidad > 0 y rinde la misma utilidad esperada que las demás en soporte.
- LH hace un proceso de *pivoting* en un **conjunto poliedrico** (el conjunto de pares de estrategias mixtas factibles) hasta encontrar un **"punto completamente etiquetado"**, que corresponde a un equilibrio de Nash.

De forma muy resumida, la intuición es:

1. Empezar con un **equilibrio trivial** (por ejemplo, un jugador elige con prob. 1 una estrategia pura) en un "juego modificado" o "artificial".
2. Realizar pivotajes que "cambian" qué estrategia pura está en la mezcla soporte.
3. Se avanza por aristas en el politopo en busca de un "punto" en el cual cada estrategia en soporte de un jugador hace que el otro sea indiferente a cierto subconjunto de sus estrategias, y viceversa.
4. Ese punto corresponde a un equilibrio en el juego original.

---

3. Pseudocódigo Esquemático de Lemke–Howson

El **código** a menudo se expresa en términos de "poliedros primales y duales" y etiquetados. A continuación, un pseudocódigo más computacional al estilo Python:


```python
def lemke_howson(U1, U2):
    """
    Implementación del algoritmo Lemke-Howson para encontrar equilibrios de Nash.

    Args:
        U1: Matriz de pagos del Jugador 1 (m x n)
        U2: Matriz de pagos del Jugador 2 (m x n)

    Returns:
        tuple: (x, y) donde x es la estrategia mixta del Jugador 1 y
               y es la estrategia mixta del Jugador 2
    """
    # Dimensiones del juego
    m, n = U1.shape

    # Paso 1: Construir representación poliédrica
    # Crear etiquetas para cada faceta
```

```
    # Las primeras m etiquetas corresponden a estrategias del Jugador 1
    # Las siguientes n etiquetas corresponden a estrategias del Jugador
2
    labels = list(range(m + n))

    # Inicializar estructuras para almacenar restricciones activas
    # active_labels guarda qué restricciones están "ajustadas"
    active_labels = set()

    # Paso 2: Iniciar en una solución trivial
    # Elegir una etiqueta k arbitraria para comenzar (típicamente k = 0)
    k = 0
    active_labels.add(k)

    # Inicializar estrategias (inicialmente, una estrategia pura)
    x = [0] * m
    y = [0] * n

    if k < m:
        x[k] = 1.0  # Jugador 1 juega estrategia pura k
    else:
        y[k - m] = 1.0  # Jugador 2 juega estrategia pura k-m

    # Paso 3: Proceso de pivotaje
    while True:
        # Encontrar etiqueta duplicada
        duplicate_label = find_duplicate_label(active_labels, labels)

        if duplicate_label is None:
            # Si no hay etiqueta duplicada, estamos en un equilibrio
            break

        # Realizar pivote para eliminar esta etiqueta duplicada
        # Esto implica moverse a lo largo de una arista en el poliedro
        new_vertex, new_active_labels = pivot(active_labels,
duplicate_label, U1, U2)

        # Actualizar estado
        active_labels = new_active_labels

        # Verificar si hemos completado un ciclo (llegado al equilibrio)
        if is_completely_labeled(active_labels, m, n):
            # Reconstruir (x, y) a partir de las etiquetas activas
            x, y = reconstruct_strategies(active_labels, U1, U2)
            break

    # Paso 5: Reconstruir las estrategias mixtas finales
    # Las estrategias mixtas son reconstruidas a partir del conjunto
final
    # de etiquetas activas (variables básicas en la terminología de PL)
    x, y = normalize_strategies(x, y)

    return x, y
```

```python
def find_duplicate_label(active_labels, all_labels):
    """Encuentra una etiqueta que aparece más de una vez en el conjunto activo"""
    # Implementación específica
    pass

def pivot(active_labels, label_to_remove, U1, U2):
    """Realiza un pivote eliminando una etiqueta e introduciendo otra"""
    # Similar a un paso del método simplex
    pass

def is_completely_labeled(active_labels, m, n):
    """Verifica si el conjunto actual de etiquetas define un equilibrio"""
    # Un conjunto es completamente etiquetado si contiene exactamente
    # una etiqueta de cada tipo (estrategia) de cada jugador
    pass

def reconstruct_strategies(active_labels, U1, U2):
    """Reconstruye las estrategias mixtas a partir de las etiquetas activas"""
    # Resolver sistema de ecuaciones para obtener probabilidades
    pass

def normalize_strategies(x, y):
    """Normaliza vectores de estrategia para asegurar que suman 1"""
    x_sum, y_sum = sum(x), sum(y)

    if x_sum > 0:
        x = [xi/x_sum for xi in x]

    if y_sum > 0:
        y = [yi/y_sum for yi in y]

    return x, y
```


Notas:

- En la práctica, se requiere llevar un registro de cuáles restricciones están activas (las que definen los valores exactos de las utilidades de estrategias en soporte).
- Lemke–Howson es un método determinista (dado el "label" de inicio) que **garantiza** encontrar *un* equilibrio de Nash.
- A veces hay varios equilibrios y el método depende de la estrategia de *pivoting* inicial para seleccionar uno.

---

4. Ejemplo Ilustrativo

Consideremos un **juego 2x2** con payoff matrices:

$U^1 = \begin{pmatrix}
2 & 0 \
1 & 1
\end{pmatrix},
\quad
U^2 = \begin{pmatrix}
1 & 0 \
0 & 2
\end{pmatrix}.$

- Jugador 1 (fila) tiene estrategias puras F_1, F_2.
- Jugador 2 (columna) tiene estrategias puras C_1, C_2.

Interpretemos los pagos:

- Si (F1, C1): Jugador 1 obtiene 2, Jugador 2 obtiene 1.
- Si (F1, C2): Jugador 1 obtiene 0, Jugador 2 obtiene 0.
- Si (F2, C1): Jugador 1 obtiene 1, Jugador 2 obtiene 0.
- Si (F2, C2): Jugador 1 obtiene 1, Jugador 2 obtiene 2.

4.1. Buscando un equilibrio

Equilibrios puros

Revisemos si hay equilibrios en puras:

- (F1, C1): Si Jugador 2 juega C1, Jugador 1 prefiere F1 (2 en vez de 1). Bien para Jugador 1. Pero Jugador 2, dado que Jugador 1 juega F1, prefiere C1 (1 vs 0). **(F1, C1) es un EN** en puras.
- (F2, C2): Si Jugador 1 juega F2, Jugador 2 prefiere C2 (2 vs 0). Jugador 1, dado que Jugador 2 juega C2, prefiere F2 (1 vs 0). **(F2, C2) es otro EN**.

Así, hay **dos equilibrios en puras**: (F1,C1) y (F2,C2).

¿Hay un equilibrio mixto distinto?

Aunque ya tenemos dos equilibrios puros, probemos si existe otro en mezclas.

- Sea x = prob de Jugador 1 en F1, (1-x) en F2.
- Sea y = prob de Jugador 2 en C1, (1-y) en C2.

Utilidad esperada de J1: $U_1(x,y) = x,y\cdot 2 + x(1-y)\cdot 0 + (1-x)y\cdot 1 + (1-x)(1-y)\cdot 1.$

Podemos hacer la **indiferencia**: Jugador 1 está indiferente entre F1 y F2 si:
$\text{Payoff}(F1) = \text{Payoff}(F2),$
dado y.

- Payoff(F1) contra y: $2y + 0(1-y) = 2y.$
- Payoff(F2) contra y: $1\cdot y + 1\cdot(1-y) = y + 1 - y = 1.$

Indiferencia => $2y = 1 \Rightarrow y = 0.5.$

Análogamente, Jugador 2 estará indiferente entre C1 y C2 si:
$\text{Payoff}(C1) = \text{Payoff}(C2),$
dado x.

- Payoff(C1): $1\cdot x + 0\cdot (1-x) = x.$
- Payoff(C2): $0\cdot x + 2\cdot (1-x) = 2 - 2x.$

Igualando => $x = 2 - 2x \Rightarrow 3x = 2 \Rightarrow x = \tfrac{2}{3}.$

Así, tenemos una posible mezcla $(x,y) = \bigl(\tfrac{2}{3}, \tfrac{1}{2}\bigr)$. Verifiquemos si realmente es un EN:

- Jugador 1: con $y=0.5$, payoff(F1)=1, payoff(F2)=1, está indiferente; jugar $\tfrac{2}{3}$ en F1 es factible.
- Jugador 2: con $x=\tfrac{2}{3}$, payoff(C1)=$\tfrac{2}{3}$, payoff(C2)=$2-2(\tfrac{2}{3})=\tfrac{2}{3}$. Indiferencia también.

Por lo tanto, **existe un tercer EN mixto**:
$\mathbf{x}^*=(\tfrac{2}{3},\tfrac{1}{3}), \quad
\mathbf{y}^*=(\tfrac{1}{2},\tfrac{1}{2}).$

4.2. Cómo Lemke–Howson lo hallaría

- Si iniciamos un pivot "forzando" que Jugador 1 juegue F1 con prob 1, en un proceso de desajuste, podríamos pivotar para "sacar" una estrategia pura y "meter" otra en soporte, etc.
- El método, tras unos cuantos pasos, podría llegar a (F1,C1) o (F2,C2) o bien al mixto $\left(\tfrac{2}{3},\tfrac{1}{3}\right), \left(\tfrac{1}{2},\tfrac{1}{2}\right)$.
- Dependiendo de la **estrategia de elección de pivote** y la "entrada artificial" inicial, se termina en *uno* de los tres equilibrios.

Conclusión: Este ejemplo de 2x2 con **2** equilibrios puros y **1** mixto muestra la característica de que **puede haber múltiples equilibrios**. Lemke–Howson (u otro algoritmo) hallará al menos uno.

PROF

5. Comentarios sobre Complejidad

- **Encontrar un EN en un juego de 2 jugadores** con payoff bimatriz de tamaño $m\times n$ está en la clase de complejidad **PPAD** (no se sabe si es NP-completo o si hay un algoritmo polinomial).
- El algoritmo **Lemke–Howson** garantiza una **solución** (un EN) pero puede requerir (en el peor caso) tiempo exponencial en $\max(m,n)$.
- Existen otros métodos (p.ej. "Support Enumeration" para juegos pequeños, métodos híbridos, etc.). Todos enfrentan la barrera de la **complejidad PPAD**: no hay (conocido) un algoritmo en tiempo polinomial que *siempre* funcione para todos los juegos en el peor caso.
- Aun así, para **juegos de tamaño moderado** (p.ej. docenas de estrategias), **Lemke–Howson** o **solvers comerciales** (como en sistemas de ecuaciones no lineales) pueden ser viables.

Conclusión

Para un **juego bimatriz** de dos jugadores, el **equilibrio de Nash** se define como un par de estrategias mixtas en las que cada jugador está jugando una mejor respuesta a la del otro. A nivel **computacional**, uno de los algoritmos clásicos es **Lemke–Howson**, que:

1. Construye la representación poliedrica de mejores respuestas.
2. Inicia con una solución trivial (estrategia pura "artificial").
3. Realiza un procedimiento de **pivot** iterativo para alcanzar un **punto completamente etiquetado**, equivalente a un EN.

Si bien **no** es polinomial en el peor caso, **garantiza** encontrar *al menos un* equilibrio para juegos 2D (2 jugadores). Para **juegos pequeños**, se puede incluso enumerar soportes o resolver las **condiciones de indiferencia** a mano (como en el ejemplo). Sin embargo, para **instancias grandes** la tarea se vuelve compleja, reflejando la **dificultad intrínseca** (PPAD) del problema de hallar un Equilibrio de Nash en juegos generales no cooperativos.