ANÁLISIS PRESCRIPTIVO: MODELOS DE OPTIMIZACIÓN

La penetración del análisis prescriptivo varia mucho entre sectores, destacando la logística, la investigación operativa, la fijación de horarios, o la estrategia de precios como áreas de mayor aplicación. Nos podemos encontrar diversos casos en los que la combinación de análisis prescriptivo y predictivo se combinen bajo un mismo objetivo, con la característica de los análisis predictivos sean elementos de entrada para los análisis prescriptivos.

1. Motivación y Objetivos

El análisis prescriptivo surge como la evolución natural del análisis descriptivo y predictivo, ya que no solo se queda en explicar o prever lo que sucede, sino que va un paso más allá: recomienda las acciones óptimas a tomar para alcanzar un objetivo específico. La idea es transformar el conocimiento obtenido a partir de los datos en estrategias y decisiones que maximicen beneficios o minimicen costos, riesgos o ineficiencias.

Ejemplo:
Imagina que eres el director de operaciones de una cadena de supermercados. Durante el análisis descriptivo, observas que ciertas tiendas tienen pérdidas en la rotación de inventario, y mediante análisis predictivo, logras prever picos de demanda en determinados productos. Sin embargo, ¿cómo decides la cantidad óptima de cada producto a enviar a cada tienda para maximizar las ventas y minimizar las pérdidas por exceso o falta de stock? Aquí es donde entra el análisis prescriptivo. Mediante modelos de optimización, defines el mejor plan logístico y de inventario que, a partir de restricciones (capacidad de almacenamiento, presupuesto, tiempos de entrega) y objetivos (maximizar las ganancias o minimizar los costos), te permite generar una recomendación concreta y operativa.

Objetivos principales del análisis prescriptivo:

- **Optimizar Recursos:** Ayudar a asignar recursos de manera eficiente, ya sea en inventarios, personal o tiempos de producción.
- **Mejorar la Toma de Decisiones:** Ofrecer una guía basada en modelos matemáticos que respalde las decisiones estratégicas, reduciendo la incertidumbre.
- **Automatizar Decisiones:** Integrar modelos de optimización en sistemas de toma de decisiones en tiempo real para que las recomendaciones se actualicen de forma dinámica ante cambios en el entorno.

En la siguiente tabla aparecen algunos casos de la analítica prescriptiva por área de aplicación y en distintos sectores

Sector	Área	Caso de uso
Finanzas	Optimización de las inversiones	Determinación de la mejor cartera en base a la tipología de inversión, riesgo y evolución del mercado
Transportes	Optimización de flotas	Programación de rotaciones de tripulaciones y aeronaves para satisfacer la planificación de vuelos
Marketing	Automatización y personalización de mensajes	Planificación y creación de flujos de envíos de correos electrónicos con el fin de maximizar la conversión de clientes
Distribución	Diseño de almacenes y rutas	Ubicación de productos en almacenes y secuencias de reparto con el fin de minimizar los tiempos de entrega
Energía	Planificación de la producción	Generación de energía eléctrica en función de la demanda teniendo en cuenta diversas fuentes y restricciones
Salud	Programación de turnos	Elaboración de calendarios de enfermería en hospitales considerando distintas condiciones y convenios

Casos de uso en la órbita de la analítica prescriptiva.

El análisis prescriptivo, por lo tanto, se convierte en una herramienta vital para empresas y organizaciones que buscan no solo comprender y anticipar el comportamiento de sus sistemas, sino también actuar de manera óptima para alcanzar sus metas.

---

2. Optimización Matemática

La optimización matemática es el corazón del análisis prescriptivo. Se encarga de formular y resolver problemas en los que se busca maximizar o minimizar una función objetivo, sujeta a un conjunto de restricciones. Este enfoque permite modelar situaciones reales de manera precisa y determinar la mejor solución posible, considerando las limitaciones inherentes al problema.

Conceptos básicos:

- **Función Objetivo:** Es la función matemática que se desea optimizar. Por ejemplo, en un problema de asignación de recursos, podría ser la maximización de las ganancias o la minimización de los costos.
- **Restricciones:** Son las limitaciones o condiciones que deben cumplirse. Estas pueden ser, por ejemplo, la capacidad de producción, los presupuestos disponibles o los tiempos de entrega.
- **Variables de Decisión:** Son las incógnitas que se deben determinar para optimizar la función objetivo. En el ejemplo de un supermercado, podrían ser las cantidades de productos a enviar a cada tienda.

Ejemplo:
Supongamos que trabajas en la planificación de la producción de una fábrica. Tu objetivo es minimizar el costo de producción manteniendo la capacidad de satisfacer la demanda. Para ello, defines la función objetivo que representa el costo total de producción, considerando tanto los costos fijos como los variables. A

continuación, incorporas restricciones tales como la disponibilidad de materias primas, la capacidad de las máquinas, el tiempo de trabajo disponible y las demandas mínimas que deben cubrirse en cada mercado. Las variables de decisión son las cantidades de productos a fabricar. Utilizando técnicas de optimización, como la programación lineal o la programación entera, encuentras la combinación exacta de producción que minimiza el costo total sin violar ninguna de las restricciones.

Métodos y Herramientas:

- **Programación Lineal (PL):**
  Se utiliza cuando tanto la función objetivo como las restricciones son lineales. Es muy común en problemas de asignación, transporte y planificación.
- **Programación Entera (PE):**
  Una variante de la PL en la que algunas o todas las variables de decisión deben ser números enteros, adecuada para problemas en los que las soluciones fraccionarias no tienen sentido (por ejemplo, asignación de personal o vehículos).
- **Algoritmos Metaheurísticos:**
  Para problemas muy complejos o no lineales, se pueden utilizar métodos como el algoritmo genético, la búsqueda tabú o el recocido simulado, que buscan soluciones aproximadas en un espacio de búsqueda muy grande.

Ejemplo de Herramientas:
Existen múltiples herramientas y lenguajes de programación para resolver problemas de optimización, tales como:

- **IBM CPLEX y Gurobi:** Solvers comerciales muy potentes para problemas de PL y PE.
- **COIN-OR:** Un conjunto de herramientas de código abierto para optimización.
- **Lenguajes de modelado:** AMPL, GAMS o incluso Python (con bibliotecas como PuLP, Pyomo o SciPy) son muy usados para formular y resolver modelos de optimización.

La optimización matemática permite, de esta forma, transformar un problema del mundo real en un modelo matemático preciso, resolverlo y, a partir de la solución, obtener recomendaciones prácticas y accionables que guían la toma de decisiones. Esta capacidad de traducir un problema complejo en un modelo resoluble es lo que hace tan valioso el análisis prescriptivo en diversas industrias.

En resumen, el análisis prescriptivo y sus modelos de optimización se fundamentan en dos pilares: la motivación de transformar datos y conocimientos en decisiones estratégicas y operativas, y el uso de la optimización matemática para encontrar la solución óptima bajo ciertas restricciones. Este enfoque permite a las organizaciones mejorar sus procesos, asignar recursos de manera eficiente y adaptarse a un entorno dinámico.

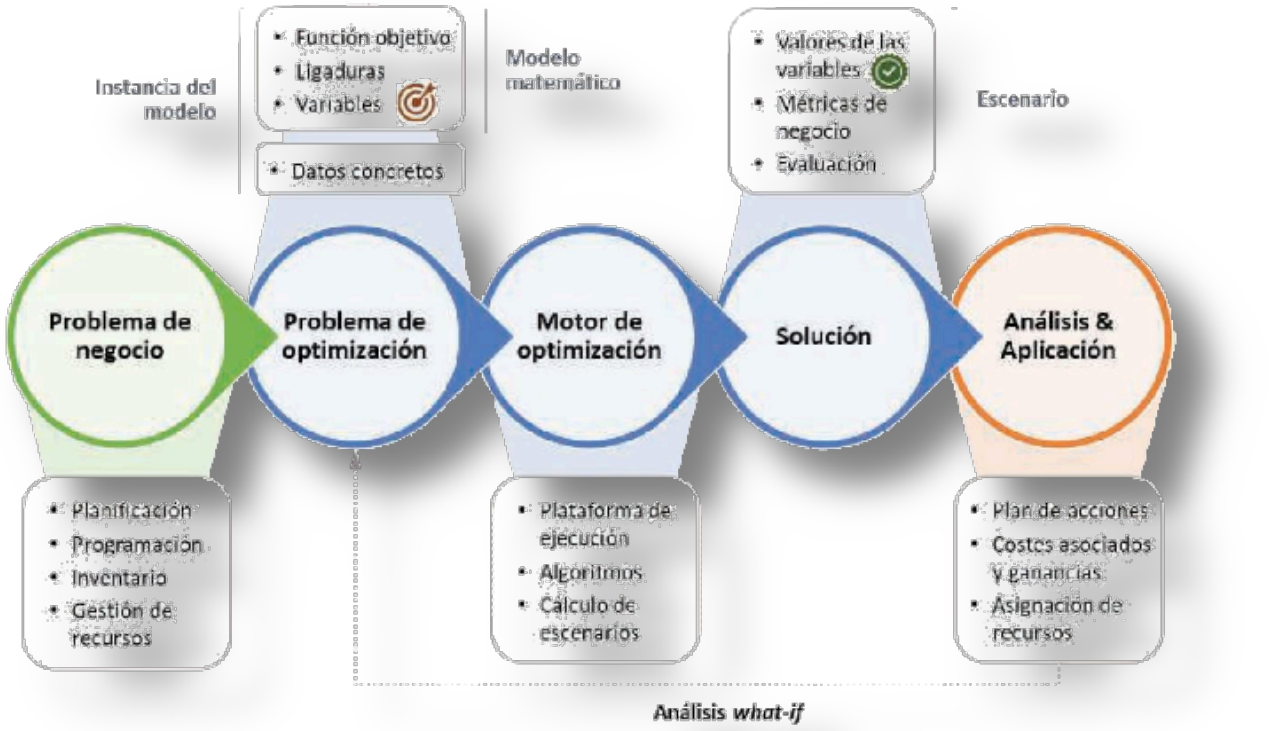


Figura 8-1. Planteamiento de un problema de optimización.

Se muestra un **planteamiento** genérico de un problema de optimización:

---

1. Definición del problema

Contexto:
Imagina una empresa que cuenta con un número limitado de máquinas y varios proyectos que deben realizarse. Cada proyecto requiere una cierta cantidad de tiempo de máquina y aporta un beneficio distinto. El objetivo es asignar las máquinas a los proyectos de forma que se maximice el beneficio total, respetando las limitaciones de disponibilidad.

Definición:
El problema consiste en determinar cuántas horas asignar a cada proyecto, de modo que se maximice el beneficio total, sin superar la capacidad total de horas de las máquinas disponibles.

---

2. Planteamiento del problema de optimización

En esta fase, se traduce el problema real a un modelo matemático. Se deben identificar las **variables de decisión**, la **función objetivo** y las **restricciones**.

Variables de decisión:
Sea x_i la cantidad de horas asignadas al proyecto i. Por ejemplo, si tenemos tres proyectos, tendremos x_1, x_2 y x_3.

Función objetivo:
Se busca maximizar el beneficio total. Si cada proyecto i aporta un beneficio de b_i por hora, la función objetivo es:

$\text{Maximizar } Z = b_1x_1 + b_2x_2 + b_3x_3$

Restricciones:

1. **Capacidad total de las máquinas:**
   Si la empresa dispone de H horas totales disponibles, se debe cumplir:

   x1+x2+x3≤Hx_1 + x_2 + x_3 \leq H

2. **Restricciones individuales por proyecto (si existen):**
   Puede haber límites máximos de horas que se pueden dedicar a un proyecto por temas de disponibilidad o requerimientos específicos. Por ejemplo, si el proyecto 1 tiene un máximo de M_1 horas:

   x1≤M1x_1 \leq M_1

   Y lo mismo para los demás proyectos.

3. **No negatividad:**
   Las horas asignadas no pueden ser negativas:

   x1,x2,x3≥0x_1, x_2, x_3 \geq 0

En conjunto, el modelo matemático queda de la siguiente forma:

$$\begin{aligned} &\text{Maximizar } Z = b_1x_1 + b_2x_2 + b_3x_3 \\ &\quad \text{sujeto a } x_1 + x_2 + x_3 \leq H, \\ &\qquad\qquad x_1 \leq M_1, \quad x_2 \leq M_2, \quad x_3 \leq M_3, \\ &\qquad\qquad x_1, x_2, x_3 \geq 0. \end{aligned}$$

---

3. Motor de optimización

Una vez formulado el problema, se utiliza un **motor de optimización** o solver para encontrar la solución óptima. Existen diferentes herramientas y software que resuelven modelos de programación lineal (y otros tipos de modelos de optimización), tales como:

- **CPLEX:** Muy utilizado en la industria para problemas de gran escala.
- **Gurobi:** Conocido por su velocidad y eficiencia en problemas lineales y enteros.
- **GLPK (GNU Linear Programming Kit):** Una opción de código abierto.
- **Solvers en Python:** Como PuLP o Pyomo, que permiten modelar y resolver problemas utilizando Python y conectarse a diferentes solvers.

En nuestro ejemplo, podríamos implementar el modelo en Python utilizando PuLP para definir las variables, la función objetivo y las restricciones, y luego utilizar un solver (por ejemplo, el de PuLP o Gurobi si se tiene acceso a él) para encontrar la asignación óptima.

---

4. Solución

El **motor de optimización** procesa el modelo y realiza la búsqueda del conjunto de valores para x_1, x_2 y x_3 que maximizan el beneficio Z, cumpliendo todas las restricciones impuestas.

Proceso:
Imagina que el motor inicia su proceso evaluando distintas combinaciones de asignación de horas. Tras varias iteraciones, identifica que asignar $x_1 = 40$ horas, $x_2 = 30$ horas y $x_3 = 20$ horas permite utilizar todas las horas disponibles de forma eficiente y maximiza el beneficio total, digamos, de $Z = \$5000$. Estos valores

representan la solución óptima según el modelo planteado.

---

5. Aplicación

Una vez obtenida la solución, es crucial interpretarla y aplicarla en el contexto real:

- **Implementación:**
  La empresa procede a asignar las máquinas y horas de operación según los valores óptimos. Se coordinan los equipos de trabajo y se programan las máquinas para que trabajen 40 horas en el proyecto 1, 30 horas en el proyecto 2 y 20 horas en el proyecto 3.
- **Verificación y validación:**
  Es importante verificar que la solución cumpla con las restricciones y se ajuste a la realidad operativa. Esto puede incluir simulaciones, análisis de sensibilidad (para ver cómo varían los resultados ante cambios en parámetros) y reuniones con el equipo operativo.
- **Retroalimentación:**
  Después de implementar la solución, se recopilan datos sobre el desempeño y se evalúa si se lograron los beneficios esperados. En caso de que el entorno operativo cambie (por ejemplo, disponibilidad de horas o beneficios por proyecto), el modelo se actualiza y se vuelve a resolver.

Resumiendo...

El proceso de formulación y resolución de un problema de optimización en investigación operativa sigue una secuencia lógica:

1. **Definir el problema** y el contexto.
2. **Plantear el modelo matemático**, estableciendo variables, función objetivo y restricciones.
3. **Seleccionar y utilizar un motor de optimización** adecuado para resolver el modelo.
4. **Obtener la solución** y analizar los resultados.
5. **Aplicar la solución** en el entorno real y ajustar el modelo según sea necesario.

En un problema de optimización, el conjunto de todas las soluciones viables, aquellas que satisfacen las ligaduras, recibe el nombre de región factible. Dentro de esta región se encuentran las soluciones optimas, que son aquellas que además dan el valor óptimo para la función objetivo. Las soluciones fuera de la región factible son las soluciones inviables, ya que violan alguna de las ligaduras establecidas.

2.1. Programación Lineal

Ejemplo:

*Tenemos una fábrica que elabora **pizzas de queso** y de **pimiento**. El beneficio que obtiene por cada unidad de **pizza de pimiento** es de **20€** y por cada **pizza de queso** es de **15€**. Para la fabricación de ambos tipos dispone de **400 horas de mano de obra** y **300 kg de harina**. Una pizza de pimiento requiere **2 horas** de mano de obra y **3 kg** de harina, mientras que para una pizza de queso son necesarias **3 horas** y **1 kg** de harina. **El objetivo es determinar las uniadades que con estos recursos se pueden fabricar de cada pizza con el fin de maximizar el margen**. Además por temas **logísticos y de conservación**, la **cantidad total de pizzas fabricadas** no puede se superior a **250 unidades**, **150** en el caso de la **pizza de pimiento**, teniendo que **servir un pedido mínimo de 50 pizzas de queso**.

Análisis paso a paso y la solución al problema utilizando técnicas de programación lineal.

---

1. Definición de variables

Sea:

- **x** = número de pizzas de **pimiento** a elaborar
- **y** = número de pizzas de **queso** a elaborar

---

2. Formulación de la función objetivo

La ganancia por cada pizza es:

- 20 € por una pizza de pimiento
- 15 € por una pizza de queso

La función a maximizar (beneficio total, Z) es:

$Z = 20x + 15y$

---

3. Planteamiento de las restricciones

La planta cuenta con dos recursos principales y se imponen condiciones adicionales:

1. **Restricción de mano de obra (horas):**
   Cada pizza de pimiento requiere 2 horas y cada pizza de queso 3 horas. Con 400 horas disponibles:

   2x+3y≤4002x + 3y \le 400
2. **Restricción de harina (kg):**
   Cada pizza de pimiento requiere 3 kg y cada pizza de queso 1 kg, con 300 kg disponibles:

   3x+y≤3003x + y \le 300
3. **Restricción de producción total:**
   La producción total no puede exceder de 250 pizzas:

   x+y≤250x + y \le 250
4. **Restricción de máximo de pizzas de pimiento:**
   No se pueden elaborar más de 150 pizzas de pimiento:

   x≤150x \le 150
5. **Restricción mínima de pizzas de queso:**
   Debe producirse al menos 50 pizzas de queso:

   y≥50y \ge 50
6. **No negatividad:**

   x≥0yy≥0x \ge 0 \quad \text{y} \quad y \ge 0

---

4. Resumen del modelo

El modelo de programación lineal queda expresado como:

Maximizar:

$$Z = 20x + 15y$$

Sujeto a:

$$\begin{cases} 2x + 3y \leq 400 & (\text{horas de mano de obra}) \\ 3x + y \leq 300 & (\text{harina}) \\ x + y \leq 250 & (\text{total de pizzas}) \\ x \leq 150 & (\text{máximo pizzas de pimiento}) \\ y \geq 50 & (\text{mínimo pizzas de queso}) \\ x,\ y \geq 0 \end{cases}$$

---

5. Análisis gráfico y búsqueda del vértice óptimo

Para identificar la solución óptima, se estudia la región factible (la intersección del conjunto de todas las restricciones) y se evalúa la función objetivo en los vértices de dicha región.

Entre las restricciones, las más restrictivas son las de la mano de obra y la harina. En efecto, se observa que:

- La restricción **2x + 3y ≤ 400** y
- La restricción **3x + y ≤ 300**
  se intersectan en un punto que suele ser candidato óptimo.

Cálculo de la intersección de 2x + 3y = 400 y 3x + y = 300

Resolver el sistema:

$$\begin{cases} 2x + 3y = 400 \\ 3x + y = 300 \end{cases}$$

1. Despejamos y de la segunda ecuación:

   y=300−3xy = 300 - 3x

2. Sustituimos en la primera:

   2x+3(300−3x)=400⟹2x+900−9x=4002x + 3(300 - 3x) = 400 \quad \Longrightarrow \quad 2x + 900 - 9x = 400

   −7x=400−900=−500⟹x=5007≈71.43-7x = 400 - 900 = -500 \quad \Longrightarrow \quad x = \frac{500}{7} \approx 71.43

3. Sustituimos x en $y = 300 - 3x$:

   y=300−3(5007)=300−15007=2100−15007=6007≈85.71y = 300 - 3\left(\frac{500}{7}\right) = 300 - \frac{1500}{7} = \frac{2100 - 1500}{7} = \frac{600}{7} \approx 85.71

Este punto:

$$(x,\ y) \approx (71.43,\ 85.71)$$

cumple además las demás restricciones:

- **x + y ≈ 157.14** (menor que 250)

- **x \le 150**
- **y \ge 50**

Evaluación de la función objetivo en el vértice

Calculamos el beneficio:

$$Z = 20\left(\frac{500}{7}\right) + 15\left(\frac{600}{7}\right) = \frac{10000}{7} + \frac{9000}{7} = \frac{19000}{7} \approx 2714.29€$$

---

6. Conclusiones y comentario sobre la solución

- **Producción óptima (en el modelo continuo):**
  - **Pizzas de pimiento:** $\frac{500}{7} \approx 71.43$ unidades
  - **Pizzas de queso:** $\frac{600}{7} \approx 85.71$ unidades
- **Beneficio máximo:**
  Aproximadamente **2714.29 €**.

> **Nota:** En la práctica, al tratarse de pizzas, la solución debería redondearse a números enteros. Una aproximación razonable podría ser producir 71 o 72 pizzas de pimiento y 85 o 86 de queso, verificando que se cumplan las restricciones. De igual forma, en problemas reales de programación entera se buscará la solución entera óptima.

Recapitulando...

El problema se formuló definiendo las variables de producción, la función de beneficio a maximizar y un conjunto de restricciones asociadas a los recursos y condiciones de producción. Resolviendo gráficamente (o analíticamente) se encontró que el óptimo se alcanza en el vértice de la región factible dado por la intersección de las restricciones de mano de obra y harina, resultando en una producción de aproximadamente 71.43 pizzas de pimiento y 85.71 pizzas de queso, con un beneficio máximo cercano a 2714.29 €.

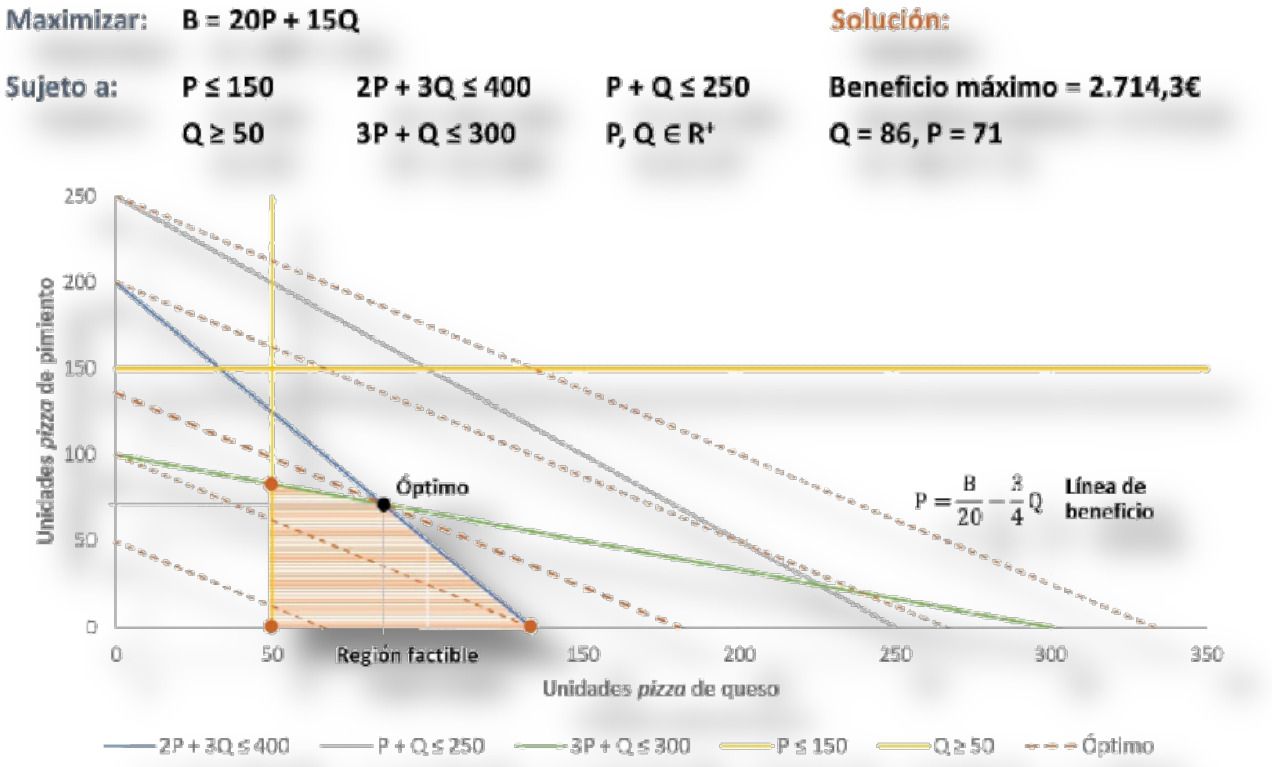


Figura 8-2. Maximización del beneficio en la fabricación de pizzas.

Explicación de la gráfica que tenemos en la parte superior:

---

1. Ejes del diagrama

- **Eje horizontal (x):** representa la cantidad de pizzas de **queso** (Q).
- **Eje vertical (y):** representa la cantidad de pizzas de **pimiento** (P).

El objetivo del problema es **maximizar el beneficio** (B = 20P + 15Q), sujetándonos a las restricciones de recursos y limitaciones logísticas (horas de mano de obra, harina, límites de producción, pedidos mínimos, etc.).

---

2. Líneas de las restricciones

Las rectas dibujadas en la gráfica corresponden a la "versión de igualdad" de cada restricción. Es decir, si una restricción se expresa como:

$$\text{Restricción:} \quad \alpha \cdot P + \beta \cdot Q \leq \gamma,$$

en la gráfica vemos la línea:

$$\alpha \cdot P + \beta \cdot Q = \gamma,$$

que marca la frontera. **Toda la zona por debajo o a la izquierda (en caso de ser $\leq$)** queda dentro de la región factible.

En concreto, se ven las siguientes líneas:

1. **2P + 3Q = 400**
   Corresponde a la restricción de **horas de mano de obra**:

   2P+3Q≤400.2P + 3Q \le 400.

   - **Zona válida:** todo lo que queda **por debajo** (o sobre la línea misma).
2. **3P + Q = 300**
   Corresponde a la restricción de **harina**:

   3P+Q≤300.3P + Q \le 300.

   - **Zona válida:** todo lo que queda **por debajo** (o sobre la línea misma).
3. **P + Q = 250**
   Limita la **producción total** de pizzas:

   P+Q≤250.P + Q \le 250.

   - **Zona válida:** todo lo que queda **por debajo** (o sobre la línea misma).
4. **P = 150**
   Marca el **máximo de pizzas de pimiento** que se puede producir:

   P≤150.P \le 150.

   - **Zona válida:** todo lo que queda a la **izquierda** de esta recta vertical (pues es un valor fijo de P sobre el eje vertical).
5. **Q = 50**
   Marca el **mínimo de pizzas de queso** requerido:

   Q≥50.Q \ge 50.

   - **Zona válida:** todo lo que queda a la **derecha** de esta recta vertical (en el eje horizontal).

Cada línea aparece con un color o trazo diferente; en la gráfica se ve cómo, al superponer todas, aparece un polígono (la **región factible**) donde se cumplen **todas** las restricciones simultáneamente.

---

3. Región factible

El área sombreada/coloreada dentro del diagrama (o marcada con un contorno) es la **región factible**. Cualquier punto dentro (o en el borde) de esa región representa una combinación (P, Q) que cumple **todas las restricciones** (mano de obra, harina, límite total, etc.). Las combinaciones de producción que caen fuera de esa región no son viables.

---

4. Punto óptimo (Óptimo)

La marquita o punto señalado como "Óptimo" indica la solución que **maximiza el beneficio**. En la gráfica:

- Se ve que está en la **intersección** de dos restricciones (límites de recursos, normalmente).
- Matemáticamente, se obtuvo resolviendo

  {2P+3Q=4003P+Q=300\begin{cases} 2P + 3Q = 400 \ 3P + Q = 300 \end{cases}

  que da $P = \frac{500}{7} \approx 71.43$ y $Q = \frac{600}{7} \approx 85.71$.

El beneficio máximo se representa a un lado, con un valor de $ \approx 2714.29 €$. Al tratarse de un problema de pizzas (unidades indivisibles), en la práctica se ajusta a números enteros cercanos, sin violar restricciones.

---

5. Interpretación final

- **Cada línea** es la versión de igualdad de las restricciones.
- **La región factible** (representada en el gráfico como un polígono sombreado) es donde se cumplen todas las desigualdades.
- **El punto "Óptimo"** se localiza en uno de los vértices de dicho polígono (por la naturaleza de los problemas de programación lineal).
- Desde ese punto, se traza la línea de isobeneficio (o líneas de nivel de la función objetivo), que se mueve hasta el último punto que aún toca la región factible, indicando la solución con el mayor valor posible de la función objetivo.

Así, la gráfica describe visualmente por qué la mejor decisión de cuántas pizzas de pimiento y queso fabricar para lograr **el beneficio máximo** se encuentra en esa intersección particular de restricciones.

Por si quieres verlo como se resuelve en Python:

Ejemplo Completo en Python

```
# Primero, asegúrate de instalar la librería PuLP si aún no la tienes instalada.
# Puedes instalarla mediante pip:
# pip install pulp

import pulp as pl

# 1. Crear el problema: Se trata de un problema de maximización
problema = pl.LpProblem("Maximizar_Beneficio_Pizzas", pl.LpMaximize)

# 2. Definir las variables:
# x: número de pizzas de pimiento (0 <= x <= 150)
# y: número de pizzas de queso (y >= 50)
x = pl.LpVariable("pizzas_pimiento", lowBound=0, upBound=150, cat="Continuous")
y = pl.LpVariable("pizzas_queso", lowBound=50, cat="Continuous")

# Nota: Si se desea que los valores sean enteros, se puede cambiar "cat" a "Integer".

# 3. Definir la función objetivo: Maximizar Z = 20x + 15y
problema += 20 * x + 15 * y, "Beneficio_Total"

# 4. Agregar las restricciones al modelo:

# Restricción 1: Mano de obra -> 2x + 3y ≤ 400
problema += 2 * x + 3 * y <= 400, "Restriccion_Mano_de_Obra"

# Restricción 2: Harina -> 3x + y ≤ 300
problema += 3 * x + y <= 300, "Restriccion_Harina"

# Restricción 3: Producción total -> x + y ≤ 250
problema += x + y <= 250, "Restriccion_Total_Produccion"

# Restricción adicional: (ya se impuso x ≤ 150 en la definición de x)
# La restricción de y ≥ 50 se impuso en la definición de y
```

```python
# 5. Resolver el problema
problema.solve()

# 6. Mostrar los resultados

print("Estado de la solución:", pl.LpStatus[problema.status])
print(f"Cantidad óptima de pizzas de pimiento: {x.varValue:.2f}")
print(f"Cantidad óptima de pizzas de queso: {y.varValue:.2f}")
print(f"Beneficio máximo obtenido: {pl.value(problema.objective):.2f} €")
```

Explicación del Código

1. **Importación y configuración:**
   - Se importa la biblioteca `pulp`.
   - Se define el problema como de maximización mediante `pl.LpProblem`.
2. **Definición de variables:**
   - Se crean dos variables: `x` (pizzas de pimiento) con límite superior de 150 y `y` (pizzas de queso) con un límite inferior de 50. Esto refleja las restricciones del problema.
3. **Función objetivo:**
   - Se define la función a maximizar: $Z = 20x + 15y$.
4. **Restricciones:**
   - **Mano de obra:** $2x + 3y \leq 400$.
   - **Harina:** $3x + y \leq 300$.
   - **Producción total:** $x + y \leq 250$.
   - Además, la restricción $x \leq 150$ y $y \geq 50$ se establecen en la definición de las variables.
5. **Solución:**
   - Se llama a `problema.solve()` para que PuLP encuentre la solución óptima.
   - Finalmente, se imprimen el estado de la solución, los valores óptimos para las variables y el beneficio máximo obtenido.

Ejecución y Resultados

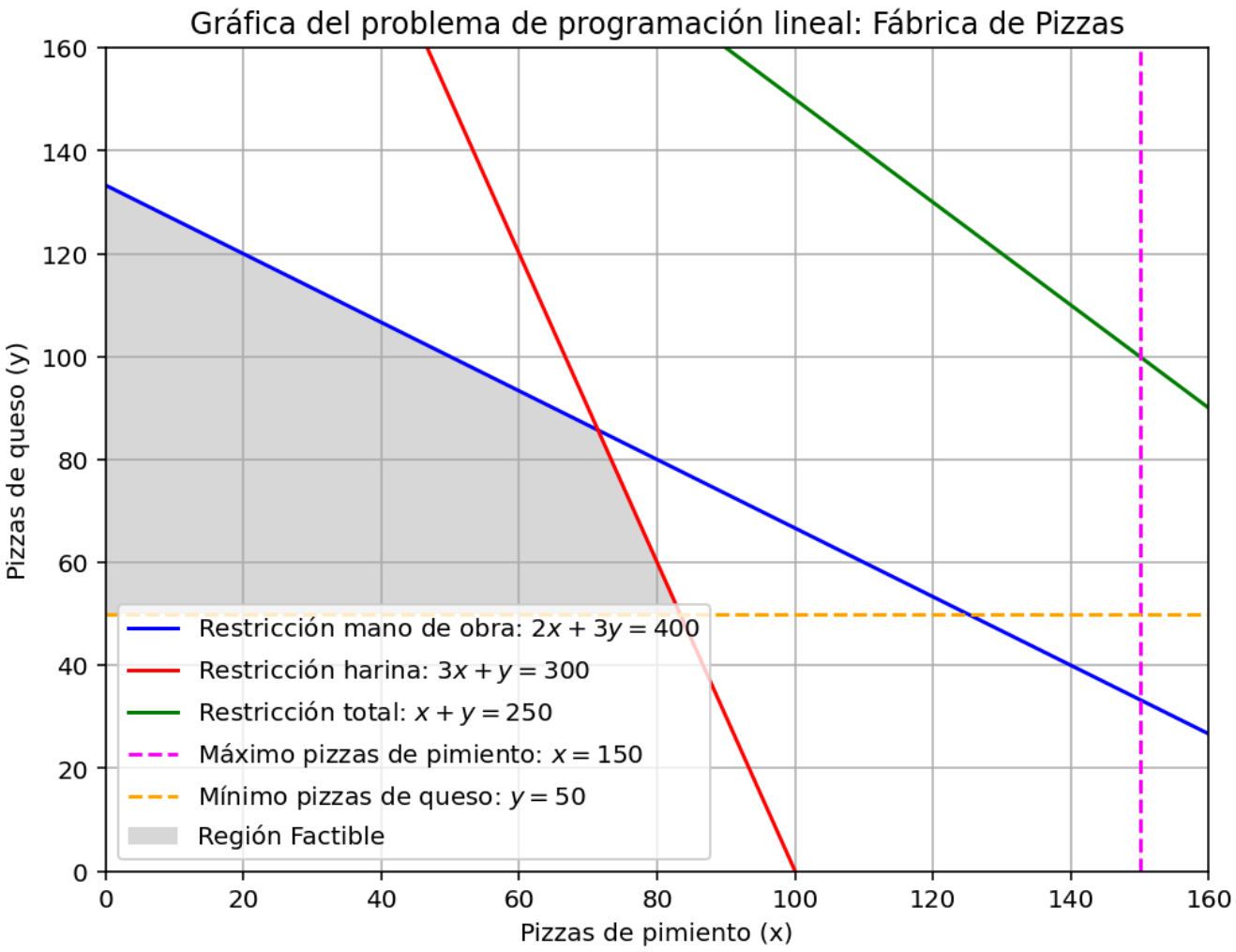
Al ejecutar este código, PuLP usará un solucionador (por defecto **CBC**) para determinar la cantidad óptima de cada tipo de pizza y el beneficio máximo. En el ejemplo analizado, la solución óptima es aproximadamente:

- **Pizzas de pimiento:** ≈ 71.43 unidades
- **Pizzas de queso:** ≈ 85.71 unidades
- **Beneficio máximo:** ≈ 2714.29 €

> **Nota:**
> En la práctica, si deseas que las soluciones sean números enteros, puedes cambiar el parámetro `cat` a `"Integer"` en la definición de las variables (`x` e `y`). Esto, sin embargo, puede hacer que el problema se vuelva de programación entera, el cual puede requerir más tiempo de cómputo para resolverse.

Con este ejemplo, se demuestra cómo modelar y resolver un problema de programación lineal en Python utilizando la librería PuLP.

Esta es la representación mediante la Python:



2.2. Otros métodos de optimización matemática

A continuación, se presentan algunos modelos de optimización alternativos, cada uno con sus características, aplicaciones y ejemplos ilustrativos:

---

1. Programación No Lineal (PNL)

Descripción

La programación no lineal se utiliza cuando la función objetivo y/o las restricciones presentan relaciones no lineales. Esto quiere decir que, a diferencia de la programación lineal, las ecuaciones o inecuaciones pueden incluir términos cuadráticos, exponenciales, logarítmicos, etc.

Aplicaciones

- **Optimización de portafolios:** Al modelar la relación riesgo-retorno, se suele utilizar la varianza (una función cuadrática) para medir el riesgo.
- **Diseño de ingeniería:** Optimización de formas, estructuras o procesos donde las propiedades físicas tienen comportamientos no lineales.

- **Problemas de equilibrio:** Donde la interacción entre variables no es simplemente aditiva.

Ejemplo

Supongamos que deseamos maximizar el beneficio de una empresa que produce dos productos, pero donde el beneficio depende de forma no lineal de la cantidad producida. Una formulación simplificada podría ser:

$$\text{Maximizar: } Z = 20x + 15y - 0.1x^2 - 0.1y^2$$

Sujeto a:

$$\begin{cases} 2x + 3y \leq 400 & (\text{restricción de recursos}) \\ x + y \leq 250 & (\text{restricción de producción total}) \\ x, y \geq 0 \end{cases}$$

En este modelo, los términos $-0.1x^2$ y $-0.1y^2$ reflejan penalizaciones (por ejemplo, por efectos de rendimientos decrecientes).
Para resolverlo se pueden utilizar algoritmos como la relajación secuencial o solvers especializados en PNL (por ejemplo, IPOPT).

---

2. Programación en Base a Restricciones (Constraint Programming, CP)

Descripción

En la programación basada en restricciones el problema se modela declarando variables, dominios y restricciones lógicas o aritméticas que deben cumplir dichas variables. En lugar de optimizar una función objetivo (aunque puede incluirse), el enfoque se centra en **explorar el espacio de soluciones** y encontrar aquellas que satisfacen todas las restricciones.
Es especialmente útil cuando el problema tiene una estructura discreta o combinatoria.

Aplicaciones

- **Problemas de asignación y horarios:** Como la asignación de turnos en hospitales o la planificación de clases en centros educativos.
- **Rompecabezas y juegos:** Sudoku, rompecabezas, y otros problemas de satisfacción de restricciones.
- **Planificación y secuenciación:** En logística y manufactura cuando se deben coordinar múltiples eventos o actividades.

Ejemplo

Considera el problema de asignar turnos a empleados. Se tienen n empleados y m turnos. Cada turno requiere un cierto número de empleados y cada empleado tiene restricciones (por ejemplo, máximo de turnos por semana, días libres, etc.).
Un modelo CP podría declararse de la siguiente manera (utilizando, por ejemplo, la librería `python-constraint` o el módulo de CP en OR-Tools):

- **Variables:** x_{ij} = 1 si el empleado i trabaja en el turno j, 0 en caso contrario.
- **Dominios:** $x_{ij} \in \{0, 1\}$.
- **Restricciones:**
  - Para cada turno j, se debe asignar un número mínimo (y máximo) de empleados.
  - Para cada empleado i, se impone un límite en el número de turnos asignados.

- Se pueden agregar restricciones adicionales, por ejemplo, que un empleado no trabaje turnos consecutivos.

El CP se encarga de recorrer las combinaciones posibles utilizando técnicas de búsqueda y propagación de restricciones, sin necesidad de plantear una función lineal u objetivo único, aunque eventualmente se puede optimizar algún criterio (por ejemplo, minimizar los costos).

---

3. Modelos Estocásticos

Descripción

La optimización estocástica se ocupa de problemas en los que algunos parámetros (como la demanda, precios, tiempos o recursos disponibles) son inciertos y se modelan mediante variables aleatorias. La formulación incorpora escenarios, distribuciones probabilísticas o incluso formulaciones de dos etapas (aquí y en el futuro) para capturar esta incertidumbre.

Aplicaciones

- **Planificación de inventarios:** Considerando la incertidumbre en la demanda, se busca minimizar los costos de almacenaje y ruptura de stock.
- **Optimización de la cadena de suministro:** Donde factores como retrasos, fallos o variaciones en los suministros se modelan estocásticamente.
- **Decisiones de inversión:** Donde los rendimientos futuros tienen incertidumbre y se utilizan formulaciones robustas o basadas en escenarios.

Ejemplo

Imagina que una empresa debe planificar la producción teniendo en cuenta la demanda incierta de sus productos. Una formulación de dos etapas podría ser:

1. **Primera etapa:** Decidir la cantidad de producción antes de conocer la demanda exacta.
2. **Segunda etapa (decisión de rescate):** Luego, una vez que se conoce la demanda (según distintos escenarios), se ajusta mediante compras adicionales o mediante penalizaciones por excedentes o faltantes.

La formulación puede ser:

$$\text{Minimizar: } C_{prod}\, x + \sum_{s \in S} p_s \left(C_{short}\, y_s^+ + C_{excess}\, y_s^- \right)$$

Sujeto a:

$$\begin{cases} x + y_s^+ - y_s^- = d_s & \forall s \in S \\ x \geq 0,\ y_s^+,\ y_s^- \geq 0 & \forall s \in S \end{cases}$$

- x: cantidad producida en la primera etapa.
- d_s: demanda bajo el escenario s con probabilidad p_s.
- y_s^+ y y_s^-: variables de rescate para ajustar la producción a la demanda (exceso o déficit).
- C_{prod}, C_{short} y C_{excess}: costos asociados a la producción, faltantes y excesos respectivamente.

Se debe definir un conjunto de escenarios S junto con sus probabilidades que reflejen la incertidumbre en la demanda. Los solvers que trabajan con optimización estocástica (o formulaciones robustas) utilizan métodos especiales para descomponer o aproximar el problema.

Resumiendo lo visto...

Cada uno de estos modelos de optimización se enfoca en distintos aspectos y tipos de problemas:

- **Programación No Lineal:** Cuando las relaciones dentro del problema no son lineales y se requiere capturar efectos de rendimientos decrecientes, aceleraciones o interacciones complejas.
- **Programación en Base a Restricciones:** Ideal para problemas de asignación, planificación y otros problemas combinatorios, donde la mayor parte de la tarea es encontrar soluciones que satisfagan una gran cantidad de restricciones lógicas o aritméticas.
- **Modelos Estocásticos:** Esenciales cuando la incertidumbre es un factor crítico en la toma de decisiones, permitiendo modelar y optimizar bajo escenarios probabilísticos.

Estos enfoques pueden incluso combinarse en problemas reales complejos, dependiendo del dominio y las necesidades específicas del análisis. Cada uno tiene sus propias técnicas, algoritmos y herramientas de software, permitiendo adaptar la formulación y solución del problema a la naturaleza particular de la situación a modelar.

---

3. ALGORITMOS GENÉTICOS

Los algoritmos genéticos (AG) son un conjunto de métodos de optimización y búsqueda inspirados en la evolución natural y la genética de los seres vivos. Utilizan conceptos como la selección, el cruzamiento (crossover) y la mutación para explorar de manera eficiente el espacio de soluciones, especialmente en problemas complejos o no lineales en los que otros métodos tradicionales podrían verse limitados.

Fundamentos de los Algoritmos Genéticos

1. Representación de Soluciones

Cada solución potencial del problema se conoce como **individuo** y generalmente se representa mediante una cadena (comúnmente un vector binario, entero, real u otra codificación) similar a un cromosoma.

2. Población

Los AG trabajan con una **población** de individuos, no con una única solución. Esto permite explorar múltiples regiones del espacio de búsqueda en paralelo.

3. Función de Aptitud (Fitness)

Se define una función de aptitud que evalúa la calidad de cada individuo. Esta función indica qué tan “buena” es una solución en términos del objetivo del problema (por ejemplo, minimizar costos o maximizar beneficios).

4. Selección

Los individuos con mayor aptitud tienen más probabilidades de ser elegidos para transmitir sus “genes” a la siguiente generación. Métodos comunes de selección son:

- **Selección por ruleta:** La probabilidad de elegir a un individuo es proporcional a su aptitud.
- **Torneo:** Se elige al mejor individuo de un grupo aleatorio.

- **Selección por ranking:** Los individuos se ordenan y se asignan probabilidades en función de su posición.

5. Cruzamiento (Crossover)

Consiste en combinar la información genética (la representación de la solución) de dos padres para crear nuevos individuos (hijos). Existen diversos métodos de cruzamiento, como:

- **Cruzamiento de un punto:** Se corta la cadena de cada padre en un punto y se intercambian las secciones.
- **Cruzamiento de dos puntos** o **uniforme:** Se intercambian segmentos o bits de forma más dispersa.

6. Mutación

Para mantener la diversidad en la población y evitar la convergencia prematura, se aplica una pequeña probabilidad de mutar (cambiar aleatoriamente) algunos genes de un individuo. Esto permite explorar nuevas áreas del espacio de soluciones.

7. Reemplazo y Evolución

Tras generar la nueva generación mediante cruzamiento y mutación, se reemplaza (total o parcialmente) a la población antigua. Este ciclo se repite durante varias iteraciones o generaciones hasta que se cumple un criterio de parada (por ejemplo, alcanzar un número máximo de generaciones o una mejora insignificante de la aptitud).

Ejemplo de un Algoritmo Genético

Imagina que deseas maximizar una función simple, por ejemplo:

$$f(x) = x \cdot \sin(10\pi x) + 1$$

donde x es un número real en el intervalo $[0, 1]$. El objetivo es encontrar el x que maximice $f(x)$.

Paso a Paso

1. **Inicialización:**
   Se genera aleatoriamente una población de, digamos, 50 individuos. Cada individuo es una posible solución, es decir, un valor de x entre 0 y 1.
2. **Evaluación:**
   Se calcula la función $f(x)$ para cada individuo. Este valor es la aptitud del individuo.
3. **Selección:**
   Se seleccionan, por ejemplo, 20 parejas de individuos (usando selección por torneo o ruleta) basándose en su aptitud para que "se reproduzcan".
4. **Cruzamiento:**
   Para cada pareja seleccionada, se toma un punto aleatorio dentro de la representación (por ejemplo, si se usa codificación binaria, un punto de corte en el cromosoma) y se mezclan los genes de los padres para crear dos hijos.
5. **Mutación:**
   Con una baja probabilidad (por ejemplo, 1% por bit o por variable), se altera aleatoriamente algunos genes de los hijos para introducir variabilidad.

6. **Reemplazo:**
   Se forma la nueva población (con los hijos y, a veces, reteniendo algunos de los mejores de la generación anterior).
7. **Criterio de Parada:**
   Se repite el proceso durante un número predefinido de generaciones o hasta que la mejora de la función objetivo sea mínima.

Al final del proceso, se espera encontrar uno o varios individuos que representan soluciones casi óptimas para el problema.

Ejemplo en Python

A continuación se presenta un ejemplo simplificado de un algoritmo genético para maximizar la función $f(x) = x \sin(10\pi x) + 1$ usando codificación en números reales:

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

# Definimos la función objetivo
def f(x):
    return x * np.sin(10 * np.pi * x) + 1

# Parámetros del algoritmo
population_size = 50
generations = 100
mutation_rate = 0.01
elite_fraction = 0.1

# Inicializamos la población con valores aleatorios en [0, 1]
population = np.random.rand(population_size)

def select(pop, fitness, num_parents):
    # Selección por ruleta
    prob = fitness / fitness.sum()
    selected = np.random.choice(pop, size=num_parents, p=prob)
    return selected

def crossover(parent1, parent2):
    # Cruzamiento de un punto (para números reales podemos promediar)
    alpha = np.random.rand()
    child = alpha * parent1 + (1 - alpha) * parent2
    return child

def mutate(x, rate):
    # Mutación: sumamos un pequeño valor gaussiano
    if np.random.rand() < rate:
        x += np.random.normal(0, 0.1)
        x = np.clip(x, 0, 1)
    return x

# Evolución del algoritmo
```

```
best_fitness = []
for gen in range(generations):
    fitness = f(population)
    best_fitness.append(np.max(fitness))

    # Conservamos una parte de la mejor población (elite)
    elite_size = int(elite_fraction * population_size)
    elite_indices = fitness.argsort()[-elite_size:]
    elite = population[elite_indices]

    # Seleccionar padres para reproducción
    num_offspring = population_size - elite_size
    offspring = []

    for _ in range(num_offspring // 2):
        parents = select(population, fitness, 2)
        child1 = crossover(parents[0], parents[1])
        child2 = crossover(parents[1], parents[0])
        offspring.extend([mutate(child1, mutation_rate),
                          mutate(child2, mutation_rate)])

    # Nueva población
    population = np.concatenate([elite, offspring])

# Resultados
best_index = np.argmax(f(population))
best_x = population[best_index]
best_y = f(best_x)
print(f"Mejor solución encontrada: x = {best_x:.4f}, f(x) = {best_y:.4f}")

# Graficamos la evolución de la aptitud
plt.plot(best_fitness)
plt.xlabel("Generación")
plt.ylabel("Mejor aptitud")
plt.title("Evolución de la aptitud")
plt.grid(True)
plt.show()
```

Explicación del Código

- **Inicialización:**
  Se crea una población aleatoria de 50 valores en $[0, 1]$.
- **Evaluación y Selección:**
  Se evalúa la función $f(x)$ para cada individuo y se utiliza la selección por ruleta para escoger padres, donde aquellos con mayor aptitud tienen mayores probabilidades.
- **Cruzamiento y Mutación:**
  Se realiza un "cruzamiento" entre dos padres mediante una combinación lineal (promedio ponderado) y se aplica una mutación (añadir ruido gaussiano) con una pequeña probabilidad.
- **Elitismo y Reemplazo:**
  Se retienen los mejores individuos (elite) y se combinan con la descendencia para formar la nueva población.

- **Ciclo Evolutivo:**
  Se repite el proceso durante un número definido de generaciones, registrando la mejor aptitud en cada generación.

Sacando conclusiones...

Los algoritmos genéticos son herramientas poderosas para abordar problemas donde la búsqueda exhaustiva o métodos tradicionales pueden ser ineficientes. Inspirados en la evolución natural, estos algoritmos exploran el espacio de soluciones a través de la combinación y variabilidad en una población de posibles soluciones. El ejemplo anterior ilustra cómo se puede implementar y aplicar un algoritmo genético para maximizar una función, pero la misma metodología puede extenderse a problemas complejos en diversas áreas, como la ingeniería, economía, logística y muchos otros campos.

4. MODELIZACIÓN PROBABILÍSTICA

La modelización probabilística es una forma de representar y analizar situaciones en las que interviene la incertidumbre. En lugar de predecir un único resultado, este enfoque trabaja con la probabilidad de que ocurra cada uno de los posibles resultados. Es decir, en vez de decir “esto va a pasar”, se dice “esto pasa con X% de probabilidad”.

A continuación, te explicaré este concepto desde lo más básico y, a modo de particularidad, profundizaremos en las cadenas de Markov, un tipo especial de modelo probabilístico.

---

1. Introducción a la Modelización Probabilística

¿Qué es y por qué utilizarla?

En la vida real, muchas situaciones no son determinísticas; es decir, no hay certeza absoluta sobre qué ocurrirá. Por ejemplo:

- El clima: Aunque podemos predecir con ciertas probabilidades si lloverá, nunca podemos asegurar con total certeza que llueva en un día específico.
- Juegos de azar: Cuando tiras un dado, el resultado es incierto, pero conocemos la probabilidad de obtener cada número (1 de 6, o aproximadamente 16.67% para cada cara).

La modelización probabilística utiliza conceptos matemáticos (como la teoría de la probabilidad) para representar esta incertidumbre. Así, en lugar de trabajar con valores fijos, usamos distribuciones de probabilidad que describen la “dispersión” de resultados posibles.

Ejemplo Básico: Lanzamiento de una Moneda

Imagina que lanzas una moneda. Los posibles resultados son “cara” o “cruz”.

- Una moneda equilibrada tiene una probabilidad de 0.5 (50%) para cada resultado.
- Si definimos una variable aleatoria X que es 1 cuando sale cara y 0 cuando sale cruz, su distribución es:

  P(X=1)=0.5yP(X=0)=0.5P(X = 1) = 0.5 \quad \text{y} \quad P(X = 0) = 0.5

Este es un ejemplo muy sencillo de modelación probabilística, ya que usamos la probabilidad para describir nuestro conocimiento sobre el resultado del lanzamiento.

2. Cadenas de Markov: Una Herramienta Específica en la Modelización Probabilística

Las cadenas de Markov son un tipo particular de modelo probabilístico que se utiliza para describir sistemas que evolucionan en el tiempo. Su característica principal es la **propiedad de Markov**, que significa que el futuro del proceso depende únicamente del estado actual y no de los estados anteriores.

Propiedad de Markov (Memoryless)

Esta propiedad se puede resumir diciendo que "el futuro es independiente del pasado, dado el presente". En términos más formales, si X_t representa el estado en el tiempo t, entonces para cualquier t y para todos los estados posibles se cumple:

$$P(X_{t+1} = j \mid X_t = i, X_{t-1} = i_{t-1}, \ldots, X_0 = i_0) = P(X_{t+1} = j \mid X_t = i)$$

Elementos de una Cadena de Markov

1. **Conjunto de Estados:**
   Es el "universo" de posibles situaciones en las que se puede encontrar el sistema. Por ejemplo, en un modelo meteorológico simple, los estados podrían ser "Soleado" y "Lluvioso".
2. **Matriz de Transición de Probabilidad:**
   Esta matriz define las probabilidades de moverse de un estado a otro. Cada elemento P_{ij} de la matriz es la probabilidad de pasar del estado i al estado j en un solo paso.
3. **Estado Inicial:**
   Especifica en qué estado se encuentra el sistema al inicio del proceso.

Ejemplo: Predicción del Clima

Imagina un modelo muy simplificado del clima en el que solo existen dos estados:

- **S** para Soleado
- **L** para Lluvioso

Supongamos que la probabilidad de que un día soleado sea seguido por otro día soleado es del 80%, y que pasa a ser lluvioso con un 20%. Además, si es lluvioso, hay un 60% de probabilidad de que el día siguiente sea lluvioso nuevamente y 40% de que se vuelva soleado. La matriz de transición P se vería así:

$$P = \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix}$$

- **Interpretación:**
  - Si hoy es soleado (estado S), la probabilidad de que mañana sea soleado es 0.8, y de que sea lluvioso es 0.2.
  - Si hoy es lluvioso (estado L), la probabilidad de que mañana sea lluvioso es 0.6, y de que se aclare a soleado es 0.4.

Evolución del Proceso

Si conocemos el estado inicial, por ejemplo, hoy está soleado, podemos predecir probabilísticamente el estado en los próximos días.

- Al día siguiente, la distribución sería: 80% soleado y 20% lluvioso.
- Para predecir dos días después, se multiplica la matriz de transición por sí misma (obteniendo la matriz de dos pasos) y se aplica a la distribución inicial.

---

3. Un Ejemplo Práctico con Cadenas de Markov

Imagina que queremos saber la probabilidad de que, en tres días, el clima sea soleado, partiendo de que hoy es soleado. Usamos la matriz de transición P descrita antes. Primero, calculamos P^3 (la tercera potencia de la matriz).

Supongamos que P^3 resulta ser:

$$P^3 = \begin{bmatrix} 0.74 & 0.26 \\ 0.52 & 0.48 \end{bmatrix}$$

Esto se interpreta de la siguiente forma:

- La primera fila de P^3 corresponde a partir de un día soleado. Así, después de tres días, si hoy es soleado, la probabilidad de que sea soleado es 0.74 y la probabilidad de que sea lluvioso es 0.26.

Este ejemplo muestra cómo las cadenas de Markov permiten modelar y predecir la evolución de un sistema en el tiempo utilizando probabilidades de transición.

---

4. Conclusión

- **Modelización Probabilística:**
  Es un enfoque que utiliza la teoría de la probabilidad para manejar situaciones inciertas. En lugar de predecir un único resultado, se trabaja con distribuciones y probabilidades de eventos.
- **Cadenas de Markov:**
  Son un tipo específico de modelización probabilística en las que el proceso evoluciona en el tiempo y tiene la propiedad de "memoria corta" (el futuro depende solo del presente).
    - **Ejemplo práctico:** Un modelo meteorológico simple en el que se predice el estado del clima (soleado o lluvioso) mediante una matriz de transición que representa las probabilidades de cambio de estado.

Estos modelos son fundamentales en diversas áreas como la economía, la ingeniería, la biología y la informática, ya que permiten analizar y predecir comportamientos complejos en sistemas estocásticos de una forma relativamente sencilla y estructurada.