Computación Evolutiva y Machine Learning: Una Perspectiva de Ingeniería Industrial

10º Congreso Internacional de Ingeniería - Instituto Tecnológico de México

Prof. D.Sc. BARSEKH-ONJI Aboud

Universidad Anáhuac México Sur

Facultad de Ingeniería

20 de octubre de 2025

Objetivos de la Presentación

- Introducir los conceptos fundamentales de la Computación Evolutiva (CE) y el Machine Learning (ML).
- Explorar cómo estas técnicas pueden transformar la práctica de la ingeniería industrial.
- Presentar ejemplos prácticos y casos de estudio relevantes.
- Discutir las implicaciones futuras y las oportunidades de investigación en este campo emergente.

Agenda

Conceptos básicos

Inteligencia Computacional (IC)

La IC es un subcampo de la Inteligencia Artificial que se enfoca en desarrollar algoritmos inspirados en procesos naturales para resolver problemas complejos. Incluye técnicas como:

- Computación Evolutiva (CE)
- Redes Neuronales Artificiales (RNA)
- Sistemas de Lógica Difusa
- Algoritmos de Enjambre
- Sistemas inmunológicos artificiales

Conceptos básicos

Inteligencia Artificial Generativa (IAG)

La IAG se refiere a sistemas de IA capaces de generar contenido nuevo y original, como texto, imágenes o música, basándose en patrones aprendidos de datos existentes. Ejemplos incluyen:

- Modelos de lenguaje grande (LLM) como GPT-4, Gemini, ... tc.
- Redes Generativas Antagónicas (GANs)

Agenda

El Imperativo de la Optimización en Ingeniería Industrial

La ingeniería industrial, en su esencia, es la disciplina de la **optimización**.

- Es el motor que impulsa la productividad, la calidad y la competitividad en cualquier sector.

El Imperativo de la Optimización en Ingeniería Industrial

La ingeniería industrial, en su esencia, es la disciplina de la **optimización**.

- Es el motor que impulsa la productividad, la calidad y la competitividad en cualquier sector.
- El ingeniero industrial es un **arquitecto de la eficiencia**, cuya misión es diseñar, mejorar e implementar sistemas integrados.

El Imperativo de la Optimización en Ingeniería Industrial

La ingeniería industrial, en su esencia, es la disciplina de la **optimización**.

- Es el motor que impulsa la productividad, la calidad y la competitividad en cualquier sector.
- El ingeniero industrial es un **arquitecto de la eficiencia**, cuya misión es diseñar, mejorar e implementar sistemas integrados.
- En el contexto actual de competencia feroz, globalización y recursos finitos, objetivos como:
  - Optimización de procesos
  - Reducción de costos
  - Mejora continua

El Imperativo de la Optimización en Ingeniería Industrial

La ingeniería industrial, en su esencia, es la disciplina de la **optimización**.

- Es el motor que impulsa la productividad, la calidad y la competitividad en cualquier sector.
- El ingeniero industrial es un **arquitecto de la eficiencia**, cuya misión es diseñar, mejorar e implementar sistemas integrados.
- En el contexto actual de competencia feroz, globalización y recursos finitos, objetivos como:
  - Optimización de procesos
  - Reducción de costos
  - Mejora continua

¡Atención!

Estos objetivos han pasado de ser opcionales a **imperativos estratégicos** para la supervivencia y el crecimiento.

Limitaciones de los Métodos de Optimización Clásicos

Históricamente, la ingeniería industrial se ha apoyado en técnicas de optimización matemática y estadística (programación lineal, cálculo de gradientes, etc.).

Fortalezas

- Extraordinariamente efectivos para problemas bien definidos.
- Ideales para relaciones lineales y funciones objetivo derivables.

Debilidades

- La realidad industrial rara vez es ordenada.
- Problemas clave (rutas, secuenciación, layout) son de optimización combinatoria **NP-duros**.
- El espacio de búsqueda crece exponencialmente, haciendo los métodos exactos computacionalmente **inviables**.
- Propensos a quedar atrapados en **óptimos locales**.

Agenda

Terminología de los Problemas de Optimización (Parte 1)

Función Objetivo (o de Costo/Aptitud): Es la función $f(x)$ que se desea minimizar o maximizar.

Terminología de los Problemas de Optimización (Parte 1)

Función Objetivo (o de Costo/Aptitud): Es la función $f(x)$ que se desea minimizar o maximizar.

Variables de Decisión: Son las variables $x = (x_1, \ldots, x_n)$ cuyos valores se buscan para optimizar la función objetivo.

Terminología de los Problemas de Optimización (Parte 1)

Función Objetivo (o de Costo/Aptitud): Es la función $f(x)$ que se desea minimizar o maximizar.

Variables de Decisión: Son las variables $x = (x_1, \ldots, x_n)$ cuyos valores se buscan para optimizar la función objetivo.

Espacio de Búsqueda: El conjunto de todos los posibles valores que pueden tomar las variables de decisión.

Terminología de los Problemas de Optimización (Parte 1)

Función Objetivo (o de Costo/Aptitud): Es la función $f(x)$ que se desea minimizar o maximizar.

Variables de Decisión: Son las variables $x = (x_1, \ldots, x_n)$ cuyos valores se buscan para optimizar la función objetivo.

Espacio de Búsqueda: El conjunto de todos los posibles valores que pueden tomar las variables de decisión.

Restricciones (Constraints): Condiciones que deben satisfacer las variables de decisión y que limitan el espacio de búsqueda a una **región factible**.

Terminología de los Problemas de Optimización (Parte 2)

Solución Factible: Un conjunto de valores para las variables de decisión que satisface todas las restricciones.

Terminología de los Problemas de Optimización (Parte 2)

Solución Factible: Un conjunto de valores para las variables de decisión que satisface todas las restricciones.

Solución Óptima: Una solución factible que produce el valor óptimo (mínimo o máximo) de la función objetivo.

Terminología de los Problemas de Optimización (Parte 2)

Solución Factible: Un conjunto de valores para las variables de decisión que satisface todas las restricciones.

Solución Óptima: Una solución factible que produce el valor óptimo (mínimo o máximo) de la función objetivo.

Óptimo Local vs. Óptimo Global: Un **óptimo local** es una solución mejor que sus vecinas. Un **óptimo global** es la mejor solución en toda la región factible.

Problema de Optimización: Maximización de Producción

- Imagina una fábrica que produce dos tipos de productos: **Producto A** y **Producto B**.
- La ganancia por unidad del Producto A es de **$40**.
- La ganancia por unidad del Producto B es de **$30**.
- Se dispone de **2400 horas** de mano de obra y **3200 kg** de materia prima.

Requisitos de producción por unidad:

Recurso	Producto A	Producto B	Disponibilidad Semanal
Mano de Obra (horas)	2	3	2400
Materia Prima (kg)	4	2	3200

Problema de Optimización: Maximización de Producción

Formulación Matemática

Sea:

- x_1 = número de unidades del Producto A a producir.
- x_2 = número de unidades del Producto B a producir.

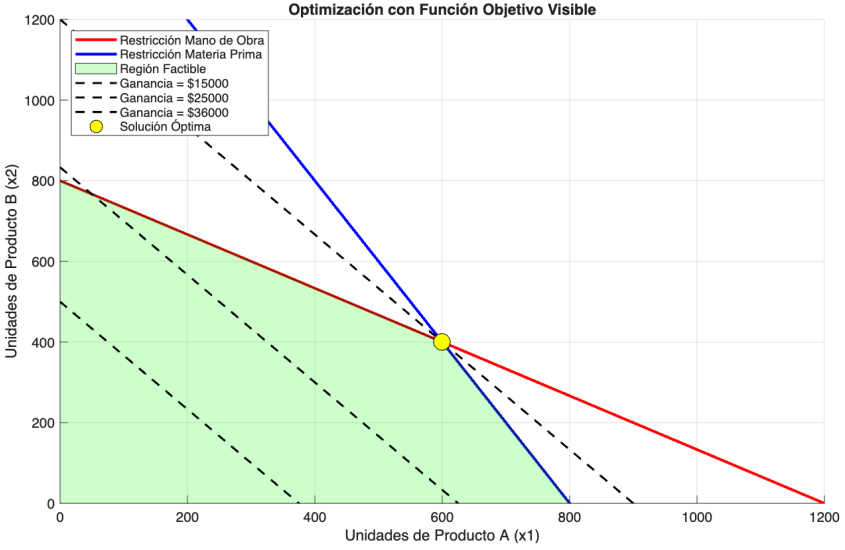
Función Objetivo (a maximizar):

$$\text{Maximizar } Z = 40x_1 + 30x_2$$

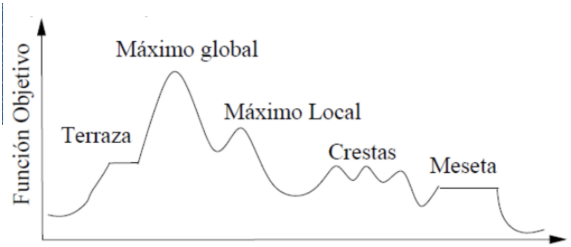
Sujeto a las siguientes restricciones:

$$2x_1 + 3x_2 \leq 2400 \quad \text{(Restricción de mano de obra)}$$
$$4x_1 + 2x_2 \leq 3200 \quad \text{(Restricción de materia prima)}$$
$$x_1, x_2 \geq 0 \quad \text{(No negatividad)}$$

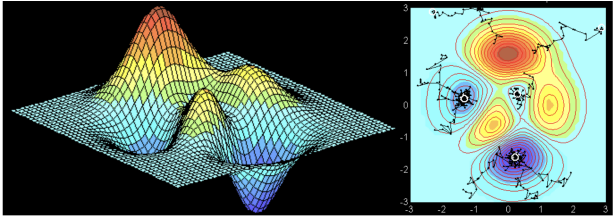
Problema de Optimización: Maximización de Producción



Tipos de Soluciones Óptimas



Espacio de Estados unidimensional



Espacio de Estados tridimensional

Figura 1: Espacio de Estados unidimensionales y tridimensionales

Modelo de Cantidad Económica de Pedido (EOQ) con Descuentos por Cantidad.

Imagina que eres el gerente de inventario de una empresa. Necesitas decidir cuántas unidades de un producto pedir a tu proveedor cada vez que haces un pedido.

- Si pides muy poco (pedidos pequeños y frecuentes): Gastarás mucho en costos de envío y procesamiento de pedidos (costo por ordenar).
- Si pides demasiado (pedidos grandes y poco frecuentes): Gastarás mucho en costos de almacenamiento, seguro y capital inmovilizado (costo de mantener inventario).

Modelo de Cantidad Económica de Pedido (EOQ) con Descuentos por Cantidad.

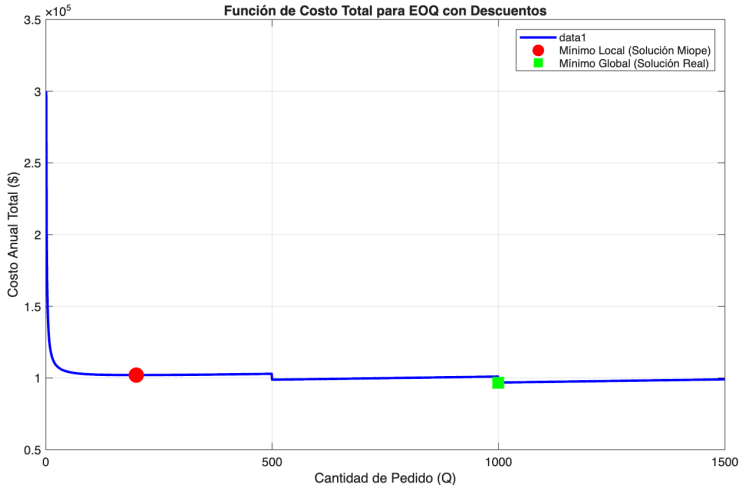
Escenario con descuentos

El modelo EOQ clásico encuentra el balance perfecto. Pero, ¿qué pasa si el proveedor te ofrece un descuento si compras más unidades?

- Comprar más podría reducir el costo total gracias al precio más bajo por unidad.
- Sin embargo, también aumentaría tus costos de almacenamiento.

Esto crea un problema de optimización complejo. La "mejor" cantidad a pedir podría ser el punto de equilibrio de un nivel de precios, o podría ser la cantidad exacta para alcanzar el siguiente nivel de descuento. Estos son los múltiples mínimos locales y el mínimo global que buscamos.

Modelo de Cantidad Económica de Pedido (EOQ) con Descuentos por Cantidad.



Presentación de Nuevos Paradigmas de IA

En respuesta a estos desafíos, han surgido dos poderosas ramas de la Inteligencia Artificial (IA):

Computación Evolutiva (CE)

- Familia de metaheurísticas estocásticas inspiradas en la **evolución biológica** de Darwin.
- Trabaja con una "población" de soluciones que "evolucionan" mediante selección, cruzamiento y mutación.
- **Fortaleza:** Explora de manera robusta y paralela espacios de búsqueda vastos y complejos, encontrando soluciones de alta calidad para problemas intratables.

Machine Learning (ML)

- Dota a los sistemas de la capacidad de **aprender patrones e inferencias directamente de los datos**, sin ser programados explícitamente.
- Construye sus propios modelos a través de la "experiencia" para predecir, clasificar y descubrir estructuras ocultas en los datos.

Tesis Central: La Sinergia Estratégica

Idea Clave

La verdadera revolución para la ingeniería industrial no reside en su aplicación aislada, sino en su **sinergia estratégica**.

- La CE no solo resuelve problemas de optimización, sino que también **potencia y automatiza el diseño de los propios modelos de Machine Learning**.
- Se crea un ciclo virtuoso:
  1. El **ML** extrae conocimiento y patrones predictivos de los datos de los sistemas industriales.
  2. La **CE** utiliza estas predicciones para optimizar las decisiones operativas (rutas, horarios, parámetros).
- Esta combinación dota al ingeniero de una capacidad sin precedentes para diseñar sistemas que no solo son eficientes, sino también **inteligentes, adaptativos y resilientes**.

Agenda

El Nuevo Rol del Ingeniero Industrial

Enfoque Tradicional:

- Analizar un sistema estático.
- Aplicar técnicas para diseñar una configuración "óptima" bajo supuestos predefinidos.
- Resultado: Soluciones **frágiles** ante la dinámica y la incertidumbre del mundo real (fallos, fluctuaciones de demanda).

Nuevo Paradigma (CE+ML):

- El objetivo ya no es "diseñar lo óptimo", sino **"diseñar sistemas que se optimizan a sí mismos"**.
- El ML permite que el sistema *aprenda* de eventos en tiempo real.
- La CE utiliza estos modelos actualizados para *re-optimizar* continuamente las operaciones.
- El ingeniero evoluciona de ser un analista a ser el **arquitecto de un sistema ciberfísico inteligente**.

Agenda

¿Qué es la Computación Evolutiva (CE)?

Inspiración: La Evolución Biológica

La Computación Evolutiva (CE) es un subcampo de la Inteligencia Artificial que se inspira directamente en los principios de la evolución biológica para desarrollar algoritmos de búsqueda y optimización.

Mecanismo General

Los algoritmos evolutivos operan sobre una **población** de soluciones candidatas, aplicando procesos de **selección**, **reproducción** y **variación** para mejorar iterativamente la calidad de las soluciones a lo largo de **generaciones**.

Antecedentes Biológicos: Primeras Ideas

Lamarckismo (Principios del s. XIX)

- Propuso el primer mecanismo coherente de evolución.
- Se basaba en la **'herencia de características adquiridas'**: los cambios en el entorno modificaban a los organismos, y estas modificaciones se transmitían a la descendencia.
- Aunque incorrecto, fue una idea revolucionaria sobre la adaptación gradual.

Selección Natural (Darwin y Wallace, 1859)

- Dentro de una población existe **variación natural**.
- Los individuos con variaciones ventajosas tienen más probabilidades de sobrevivir y reproducirse (**'supervivencia del más apto'**).
- Con el tiempo, este proceso conduce a la adaptación de las poblaciones.

La Pieza Faltante y la Síntesis Moderna

El Problema de la Herencia

La principal debilidad de la teoría de Darwin era la falta de un mecanismo de herencia. La teoría de la 'herencia por mezcla' (que sugería que las características se promediaban) contradecía la selección natural.

La Genética Mendeliana

Sin que Darwin lo supiera, Gregor Mendel demostró que la herencia es **particulada**, no por mezcla. Los rasgos se heredan a través de 'factores' discretos (hoy llamados **genes**) que no se diluyen.

La Pieza Faltante y la Síntesis Moderna

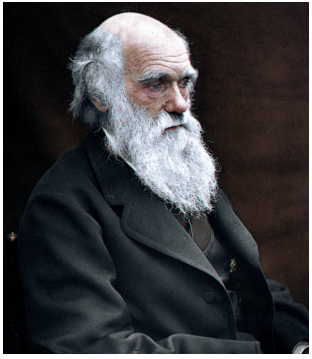


Figura 2: Charles Darwin

Figura 3: Gregor Mendel

La Pieza Faltante y la Síntesis Moderna

La Síntesis Moderna (Neo-Darwinismo)

Unificó la selección natural de Darwin con la genética mendeliana. La evolución es el resultado de la interacción de:

- **Variación Genética:** (Mutación y Recombinación).
- **Herencia:** Transmisión de genes.
- **Selección Natural:** Supervivencia diferencial.

Este es el marco conceptual que inspira directamente a la Computación Evolutiva.

Agenda

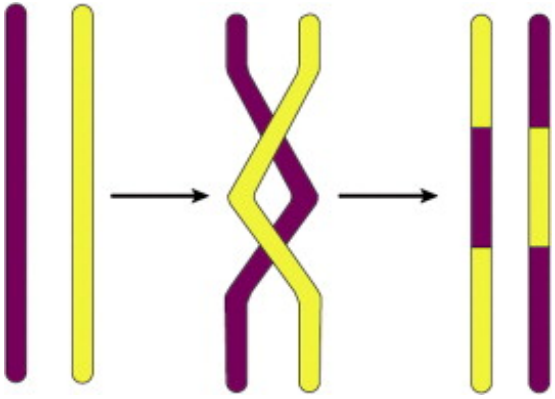


Figura 4: Cruce (Crossover)

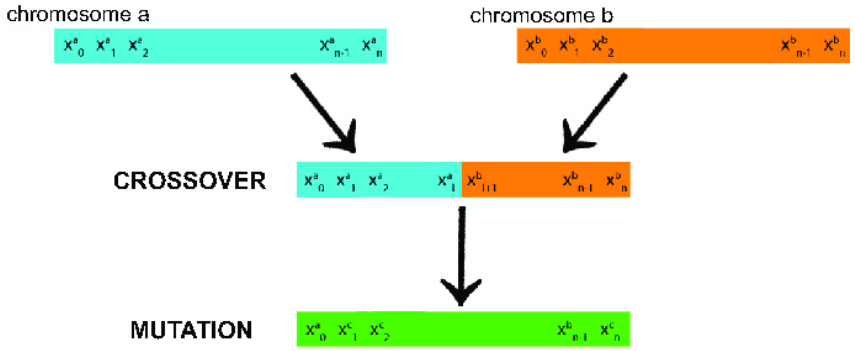


Figura 5: Cruce (Crossover) con Mutación (Mutation)

Agenda

Pioneros y Paradigmas Fundacionales

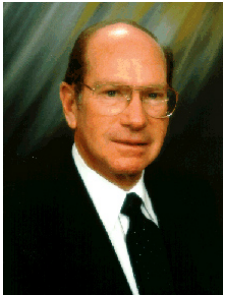
Primeros Vislumbres Computacionales

La idea de simular la evolución en una computadora es casi tan antigua como la computación misma. Visionarios como **Alan Turing** ya lo sugerían en 1950. En las décadas de 1950 y 1960, pioneros como Barricelli, Fraser y Box realizaron las primeras simulaciones de procesos evolutivos.

Los Tres Pilares de la Computación Evolutiva

En la década de 1960, surgieron de forma independiente tres líneas de investigación que se convertirían en los pilares del campo, cada una con un enfoque y una motivación distintos.

Los Tres Paradigmas Fundacionales



Programación Evolutiva (EP)

Lawrence J. Fogel (EE.UU.)

- **Motivación:** IA, crear sistemas que se adaptan y predicen.
- **Enfoque:** Evolucionar el **comportamiento** (fenotipo) de las soluciones.
- **Operador Clave: Mutación**.

Los Tres Paradigmas Fundacionales

Estrategias Evolutivas (ES)

Rechenberg y Schwefel (Alemania)

- **Motivación:** Optimización de problemas de **ingeniería** con parámetros de valor real.
- **Concepto Clave: Auto-adaptación** (el algoritmo aprende y ajusta sus propios parámetros).

Los Tres Paradigmas Fundacionales

Algoritmos Genéticos (GA)

John H. Holland (EE.UU.)

- **Motivación:** Entender teóricamente los principios de la **adaptación**.
- **Enfoque:** Énfasis en la **codificación** de la solución (genotipo).
- **Operador Clave: Cruce (Crossover)**.

Ejemplo: Paso a Paso de una Iteración (1+1)-ES

Problema

Minimizar la función de Rosenbrock: $f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$

1. **Inicialización (Padre):** Partimos de un punto inicial $\mathbf{x}^t = (-1{,}0; 1{,}0)$, con una aptitud (costo) de $f(\mathbf{x}^t) = 4{,}0$.

Ejemplo: Paso a Paso de una Iteración (1+1)-ES

Problema

Minimizar la función de Rosenbrock: $f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$

1. **Inicialización (Padre):** Partimos de un punto inicial $\mathbf{x}^t = (-1{,}0; 1{,}0)$, con una aptitud (costo) de $f(\mathbf{x}^t) = 4{,}0$.
2. **Mutación:** Se genera un nuevo individuo (hijo) añadiendo ruido Gaussiano. Supongamos que el resultado es $\mathbf{x}^{t+1} = (-0{,}39; 1{,}57)$.

Ejemplo: Paso a Paso de una Iteración (1+1)-ES

Problema

Minimizar la función de Rosenbrock: $f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$

1. **Inicialización (Padre):** Partimos de un punto inicial $\mathbf{x}^t = (-1{,}0; 1{,}0)$, con una aptitud (costo) de $f(\mathbf{x}^t) = 4{,}0$.
2. **Mutación:** Se genera un nuevo individuo (hijo) añadiendo ruido Gaussiano. Supongamos que el resultado es $\mathbf{x}^{t+1} = (-0{,}39; 1{,}57)$.
3. **Evaluación:** Se calcula la aptitud del nuevo individuo. $f(\mathbf{x}^{t+1}) = f(-0{,}39; 1{,}57) = 201{,}416$.

Ejemplo: Paso a Paso de una Iteración (1+1)-ES

Problema

Minimizar la función de Rosenbrock: $f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$

1. **Inicialización (Padre):** Partimos de un punto inicial $\mathbf{x}^t = (-1{,}0; 1{,}0)$, con una aptitud (costo) de $f(\mathbf{x}^t) = 4{,}0$.
2. **Mutación:** Se genera un nuevo individuo (hijo) añadiendo ruido Gaussiano. Supongamos que el resultado es $\mathbf{x}^{t+1} = (-0{,}39; 1{,}57)$.
3. **Evaluación:** Se calcula la aptitud del nuevo individuo. $f(\mathbf{x}^{t+1}) = f(-0{,}39; 1{,}57) = 201{,}416$.
4. **Selección:**

Como $f(\text{hijo}) > f(\text{padre})$, la nueva solución es peor. En una (1+1)-ES, el hijo se descarta y el padre sobrevive para la siguiente generación. El proceso se repite.

Agenda

Algoritmo Genético (AG)

Concepto

Popularizados por John H. Holland, los Algoritmos Genéticos (AG) son modelos que simulan la **evolución genética**. A diferencia de EP y ES, el énfasis está en el **genotipo** (la codificación de la solución).

Operadores Principales

El proceso evolutivo en un AG es impulsado principalmente por:

- **Selección:** Modela la supervivencia del más apto.
- **Cruce (Crossover):** Modela la recombinación y la reproducción sexual. Es el operador de búsqueda principal.

El Algoritmo Genético Canónico

La Propuesta Original de Holland

El AG 'canónico' o clásico se define por las siguientes características específicas:

- **Representación:** Se utiliza una **cadena de bits** (*bitstring*) de longitud fija para codificar cada solución.

El Algoritmo Genético Canónico

La Propuesta Original de Holland

El AG 'canónico' o clásico se define por las siguientes características específicas:

- **Representación:** Se utiliza una **cadena de bits** (*bitstring*) de longitud fija para codificar cada solución.
- **Selección:** Se emplea la **selección proporcional a la aptitud** (a menudo implementada como una 'rueda de ruleta') para elegir a los padres.

El Algoritmo Genético Canónico

La Propuesta Original de Holland

El AG 'canónico' o clásico se define por las siguientes características específicas:

- **Representación:** Se utiliza una **cadena de bits** (*bitstring*) de longitud fija para codificar cada solución.
- **Selección:** Se emplea la **selección proporcional a la aptitud** (a menudo implementada como una 'rueda de ruleta') para elegir a los padres.
- **Cruce (Crossover):** El **cruce de un punto** se utiliza como el método principal para producir descendencia, combinando 'bloques de construcción' de los padres.

El Algoritmo Genético Canónico

La Propuesta Original de Holland

El AG 'canónico' o clásico se define por las siguientes características específicas:

- **Representación:** Se utiliza una **cadena de bits** (*bitstring*) de longitud fija para codificar cada solución.
- **Selección:** Se emplea la **selección proporcional a la aptitud** (a menudo implementada como una 'rueda de ruleta') para elegir a los padres.
- **Cruce (Crossover):** El **cruce de un punto** se utiliza como el método principal para producir descendencia, combinando 'bloques de construcción' de los padres.
- **Mutación:** La mutación (invertir un bit al azar) se considera un operador de fondo, de menor importancia, con el único fin de mantener la diversidad genética.

Ejemplo: Algoritmo Genético para la Mochila

Problema

Seleccionar los objetos más valiosos para una mochila con capacidad de **15 kg**.

Objeto	Peso (kg)	Valor ($)
1: Linterna	4	5
2: Cuerda	7	12
3: Comida	5	8
4: Agua	6	10

- **Cromosoma:** Un vector binario de 4 bits. Un '1' significa llevar el objeto, un '0' es dejarlo.

Ejemplo: Algoritmo Genético para la Mochila

1. **Inicialización (Población Inicial):** Creamos dos soluciones (padres) al azar.
   Padre A: `[1 0 1 1]` $\rightarrow$ Peso: 15kg, Valor: \$23. (*Válido*)
   Padre B: `[0 1 1 1]` $\rightarrow$ Peso: 18kg, Valor: \$30. (*Inválido, excede 15kg*)

Ejemplo: Algoritmo Genético para la Mochila

1. **Inicialización (Población Inicial):** Creamos dos soluciones (padres) al azar.
   Padre A: `[1 0 1 1]` $\rightarrow$ Peso: 15kg, Valor: \$23. (*Válido*)
   Padre B: `[0 1 1 1]` $\rightarrow$ Peso: 18kg, Valor: \$30. (*Inválido, excede 15kg*)
2. **Crossover (Cruce de un punto):** Cruzamos a los padres después del segundo bit.
   Hijo 1: `[1 0 | 1 1]` $\rightarrow$ `[1 0 1 1]`
   Hijo 2: `[0 1 | 1 1]` $\rightarrow$ `[0 1 1 1]`

Ejemplo: Algoritmo Genético para la Mochila

1. **Inicialización (Población Inicial):** Creamos dos soluciones (padres) al azar.
   Padre A: `[1 0 1 1]` $\rightarrow$ Peso: 15kg, Valor: \$23. (*Válido*)
   Padre B: `[0 1 1 1]` $\rightarrow$ Peso: 18kg, Valor: \$30. (*Inválido, excede 15kg*)
2. **Crossover (Cruce de un punto):** Cruzamos a los padres después del segundo bit.
   Hijo 1: `[1 0 | 1 1]` $\rightarrow$ `[1 0 1 1]`
   Hijo 2: `[0 1 | 1 1]` $\rightarrow$ `[0 1 1 1]`
3. **Mutación:** Mutamos el tercer bit del **Hijo 2**.
   Hijo 2 (mutado): `[0 1 0 1]`

Ejemplo: Algoritmo Genético para la Mochila

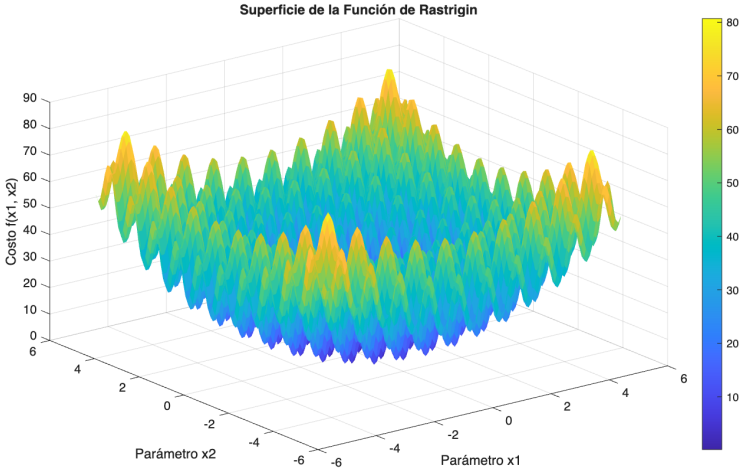
1. **Inicialización (Población Inicial):** Creamos dos soluciones (padres) al azar.
   Padre A: `[1 0 1 1]` $\rightarrow$ Peso: 15kg, Valor: \$23. (*Válido*)
   Padre B: `[0 1 1 1]` $\rightarrow$ Peso: 18kg, Valor: \$30. (*Inválido, excede 15kg*)
2. **Crossover (Cruce de un punto):** Cruzamos a los padres después del segundo bit.
   Hijo 1: `[1 0 | 1 1]` $\rightarrow$ `[1 0 1 1]`
   Hijo 2: `[0 1 | 1 1]` $\rightarrow$ `[0 1 1 1]`
3. **Mutación:** Mutamos el tercer bit del **Hijo 2**.
   Hijo 2 (mutado): `[0 1 0 1]`
4. **Evaluación (Nueva Generación):** Calculamos la aptitud de los hijos.
   Hijo 1: `[1 0 1 1]` $\rightarrow$ Peso: 15kg, Valor: \$23. (*Válido*)
   Hijo 2 (mutado): `[0 1 0 1]` $\rightarrow$ Peso: 13kg, Valor: \$22. (*Válido*)

Ejemplo: Algoritmo Genético para la Mochila

1. **Inicialización (Población Inicial):** Creamos dos soluciones (padres) al azar.
   Padre A: `[1 0 1 1]` → Peso: 15kg, Valor: \$23. (*Válido*)
   Padre B: `[0 1 1 1]` → Peso: 18kg, Valor: \$30. (*Inválido, excede 15kg*)
2. **Crossover (Cruce de un punto):** Cruzamos a los padres después del segundo bit.
   Hijo 1: `[1 0 | 1 1]` → `[1 0 1 1]`
   Hijo 2: `[0 1 | 1 1]` → `[0 1 1 1]`
3. **Mutación:** Mutamos el tercer bit del **Hijo 2**.
   Hijo 2 (mutado): `[0 1 0 1]`
4. **Evaluación (Nueva Generación):** Calculamos la aptitud de los hijos.
   Hijo 1: `[1 0 1 1]` → Peso: 15kg, Valor: \$23. (*Válido*)
   Hijo 2 (mutado): `[0 1 0 1]` → Peso: 13kg, Valor: \$22. (*Válido*)
5. **Selección:**

El **Padre A** y el **Hijo 1** son las mejores soluciones válidas hasta ahora (ambas con valor \$23). El **Hijo 2** es también una buena solución válida. El **Padre B** (inválido) se descarta. Las soluciones válidas con mayor valor sobreviven para la siguiente generación.

Aplicación del Algoritmo Genético



Aplicación del Algoritmo Genético

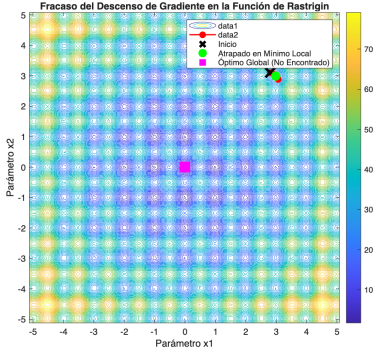


Figura 6: Fallo de los métodos tradicionales

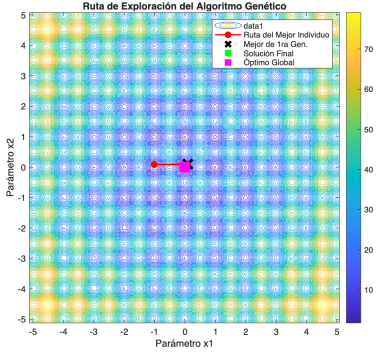


Figura 7: Éxito del Algoritmo genético