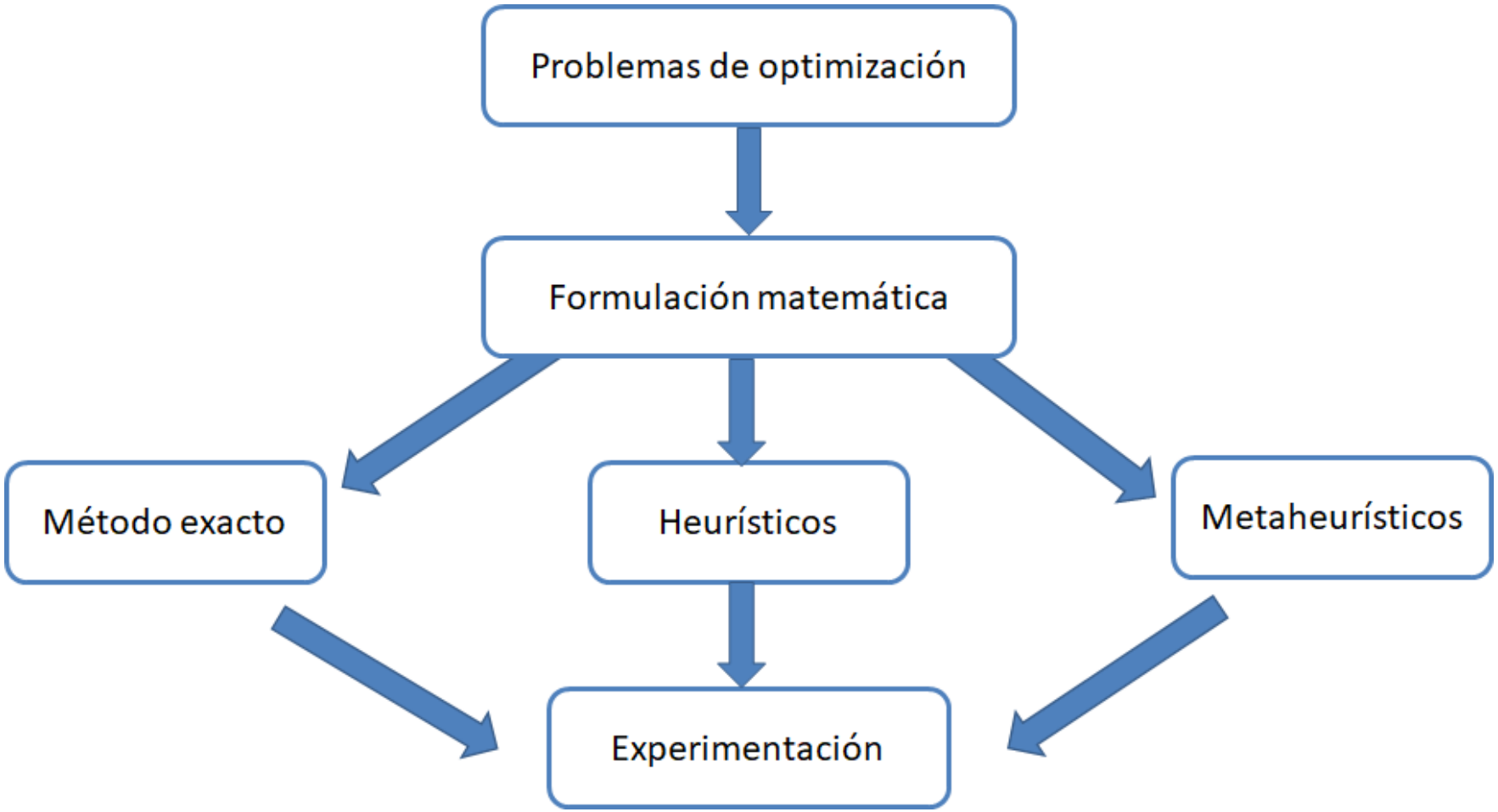
Complejidad y sistemas basados en la naturaleza

Fernando Elizalde Ramírez

Diseño de algoritmos matemáticos bioinspirados
Departamento de Matemáticas
Tecnológico de Monterrey

August 14, 2023

Introducción



Metodo heurístico

Es un procedimiento para resolver un problema complejo de optimización mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución de manera eficiente.

Problemas combinatoriales

- Transporte
- Selección de rutas
- Selección de proyectos
- Problemas de expansión
- Problemas de Control
- Priorización

Objetivo de las técnicas heurísticas combinatoriales

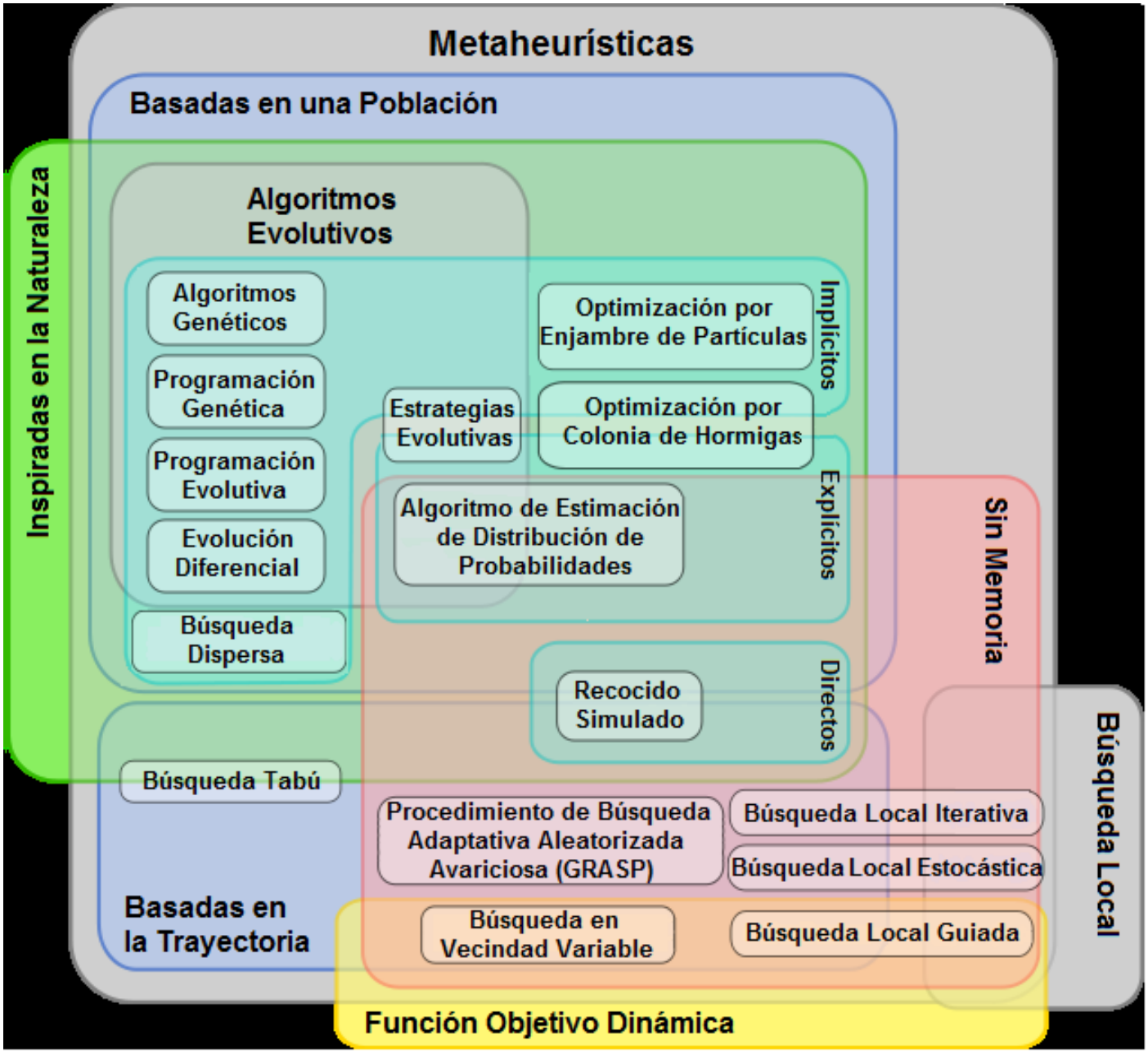
Desarrollar técnicas eficientes para encontrar un mínimo o un máximo valor de una función de muchas variables discretas independientes y con gran cantidad de soluciones, sin tener que conocer los valores del objetivo de todas esas soluciones factibles

Metaheurísticas

Estrategias que guían heurísticas a fin de encontrar una buena solución para un problema de optimización mediante el escape de óptimos locales.

Métodos Metaheurísticos de Optimización más conocidos

- Algoritmos genéticos
- Enfriamiento Simulado
- Búsqueda Tabú
- Grasp


Metaheurísticas
Basadas en una Población
Inspiradas en la Naturaleza
Algoritmos Evolutivos
Algoritmos Genéticos
Programación Genética
Programación Evolutiva
Evolución Diferencial
Búsqueda Dispersa
Estrategias Evolutivas
Optimización por Enjambre de Partículas
Optimización por Colonia de Hormigas
Implícitos
Explícitos
Algoritmo de Estimación de Distribución de Probabilidades
Sin Memoria
Recocido Simulado
Directos
Búsqueda Tabú
Procedimiento de Búsqueda Adaptativa Aleatorizada Avariciosa (GRASP)
Búsqueda Local Iterativa
Búsqueda Local Estocástica
Basadas en la Trayectoria
Búsqueda en Vecindad Variable
Búsqueda Local Guiada
Búsqueda Local
Función Objetivo Dinámica

Algoritmos bionspirados

Tratan de imitar la evolución natural de las especies y la respuesta de los sistemas sociales ante los desafíos que se les presentan. Dentro de estos, como veremos, se encuentra la familia de los algoritmos evolutivos y, más en concreto, los algoritmos genéticos, que se basan en la evolución de las especies y en la de los cromosomas respectivamente.

Características de los algoritmos bioinspirados

Los algoritmos bioinspirados se caracterizan por ser no determinísticos. Además, presentan a menudo, implícitamente, una estructura paralela y son adaptativos.

Dentro de un algoritmo bioinspirado, encontramos las siguientes partes bien diferenciadas:

- **Población**: conjunto de candidatos a ser solución.
- **Mecanismo de Evolución**: conjunto de métodos por los cuales se modifica la población.
- **Mecanismo de Calificación**: conjunto de métodos por los cuales se asigna un valor a cada elemento de la población (solución). Para asignar las calificaciones se utiliza normalmente la función a optimizar.
- **Condiciones de Parada**: procedimiento que controla si se ha encontrado una solución o se ha alcanzado un número de operaciones determinado con anterioridad.

ALGORITMOS BIOINSPIRADOS

REDES NEURONALES

ALGORITMOS INMUNOLÓGICOS

- Selección clonal
- Red inmunitaria
- Selección negativa
- Células dendríticas

INTELIGENCIA DE ENJAMBRE

- Enjambre de partículas
- Colonia de hormigas
- Colonia de abejas
- Banco de peces

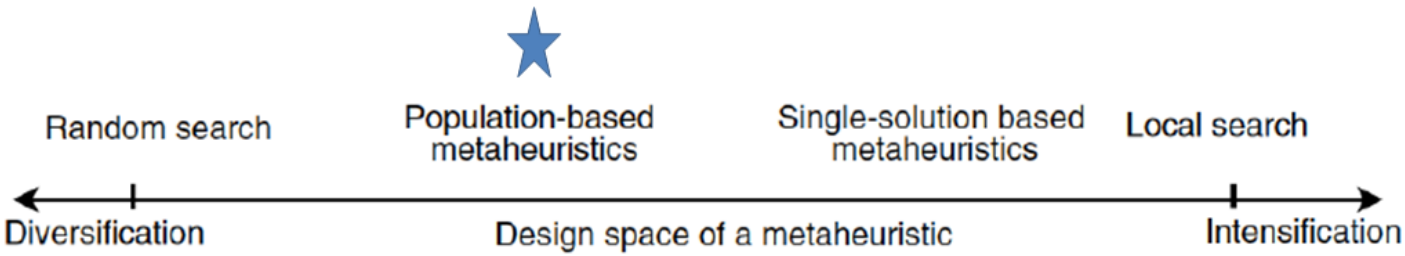
ALGORITMOS EVOLUTIVOS

- Algoritmos genéticos
- Evolución diferencial
- Estrategias evolutivas
- Programación evolutiva
- Programación genética
- Entrenamiento simulado
- Estimación de la dstribución
- PAES
  (Pareto Archived Evolution Strategy)
- NGSA-II
  (Nondominated Sorting Genetic Algorithm)

www.iic.uam.es

¿Qué resulta importante en la soluciones?

- **Intensificación**: se refiere a qué tanto nos importa la calidad de la solución a lo largo de la implementación del metaheurístico.
- **Diversificación**: se refiere a qué tanto nos importa encontrar diferentes soluciones a lo largo de la implementación del metaheurístico.



Algoritmos basados en poblaciones

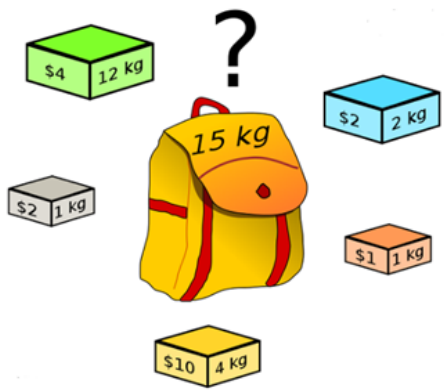
Algorithm 3.1 High-level template of P-metaheuristics.

$P = P_0$; /* Generation of the initial population */
$t = 0$;
Repeat
  Generate(P'_t); /* Generation a new population */
  P_{t+1} = Select-Population($P_t \cup P'_t$); /* Select new population */
  $t = t + 1$;
Until Stopping criteria satisfied
Output: Best solution(s) found.

Evolutivos
Búsqueda esparcida (scatter search)
Colonia de hormigas (ant colonies)
Colonia de abejas (bee colonies)
Entre otros.

Problema de la mochila

Dado un conjunto de artículos con un beneficio y capacidad, se debe decidir que artículos meter en la mochila de tal manera que se maximice el beneficio respetando la capacidad de la misma. Establece la formulación matemática. ¿Fácil o difícil?



Ejemplo 1

Un armador tiene un carguero con capacidad de hasta 700 toneladas. El carguero transporta contenedores de diferentes pesos para una determinada ruta. En la ruta actual el carguero puede transportar algunos de los siguientes contenedores:

Contenedor	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
Peso [ton]	100	155	50	112	70	80	60	118	110	55
Beneficio [$ /ton]	1741	1622	1016	1264	1305	1389	1797	1330	1559	1578

El analista de la empresa del armador desea determinar el envío (conjunto de contenedores) que maximiza el beneficio.
Resuelve a optimalidad, usando GAMS. Recopila información sobre función objetivo y tiempo computacional y el número de iteraciones.

Ejemplo 2

Al pasar del tiempo surge un nuevo pedido de embarco, teniendo como información

Contenedor	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
Peso [ton]	61	58	92	50	108	83	93	101	54	50
Beneficio [$/ton]	1100	1147	1442	1591	1078	1385	1777	1196	1753	1371
Contenedor	c_11	c_12	c_13	c_14	c_15	c_16	c_17	c_18	c_19	c_{20}
Peso [ton]	72	51	100	108	91	112	66	58	110	73
Beneficio [$/ton]	1517	1675	1193	1177	1365	1143	1314	1526	1470	1605

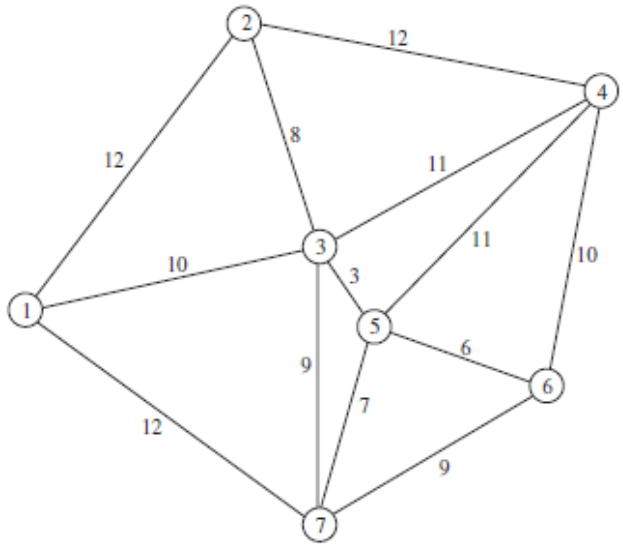
El analista de la empresa del armador desea determinar el envío (conjunto de contenedores) que maximiza el beneficio.
a) Resuelve a optimalidad, usando GAMS. Recopila información sobre función objetivo y tiempo computacional y el número de iteraciones.
b) Comparando los dos casos, ¿Qué concluyes?

Algoritmo heurístico

- Elabora un algoritmo heurístico para obtener una solución factible al problema. Presenta un pseudocódigo o diagrama de flujo.
- Implementa tu algoritmo heurístico en el ejemplo 1. Recopila información sobre función objetivo y tiempo computacional.
- Implementa tu algoritmo heurístico en el ejemplo 2. Recopila información sobre función objetivo y tiempo computacional.
- Compara los resultados obtenidos usando GAMS y los obtenidos usando tu heurístico.

Problema del agente viajero

En la figura se muestra un pequeño problema del agente viajero con siete ciudades. La ciudad 1 es el lugar de residencia del agente. Por lo tanto, si comienza desde su ciudad, el agente debe elegir una ruta para visitar cada una de las otras ciudades exactamente una vez antes de regresar a su punto de partida. Determina el orden óptimo en que debe de visitar cada una de las ciudades.



- Elabora un algoritmo heurístico para obtener una solución factible al problema. Presenta un pseudocódigo o diagrama de flujo.
- Usando el algoritmo propuesto, determine una solución del problema.

Complejidad

Es en realidad una métrica teórica que mide la eficiencia computacional de un algoritmo. Si tenemos varios algoritmos que solucionan un mismo problema, entonces esta métrica nos ayudará a definir cuál de los algoritmos es mejor en términos computacionales.
Básicamente la medición del algoritmo es cuánto tarda en resolver un problema.

Un algoritmo necesita dos importantes recursos para resolver un problema: tiempo y espacio. La complejidad de un algoritmo con respecto al tiempo computacional, se refiere a la cantidad de pasos requeridos para resolver un problema de tamaño n. En el peor de los casos la cantidad de operaciones para implementar el algoritmo.

El objetivo principal al momento de calcular la complejidad del problema no es solamente calcular el número exacto de pasos si no que se pueda establecer una cota superior.

Theorem

La función f pertenece a la clase de complejidad de g ($f \in O(g)$) si existe un c y n_0 tales que para todo $n \geq n_0$ se tiene que $|f(n)| \leq c|g(n)|$.

Intuitivamente, sólo se considera el término más importante y se ignoran los factores constantes.

Ejemplos

- $3n^2 + 5n - 7 \in O(n^2)$
- $O(2g(n)) = O(g(n))$
- $O(\log\ n) = O(\ln\ n)$
- $O(3n^2 + 5n - 7) = O(n^2)$
- $O(n^2) \subset O(n^3)$
- $O(2^n) \subset O(3^n)$

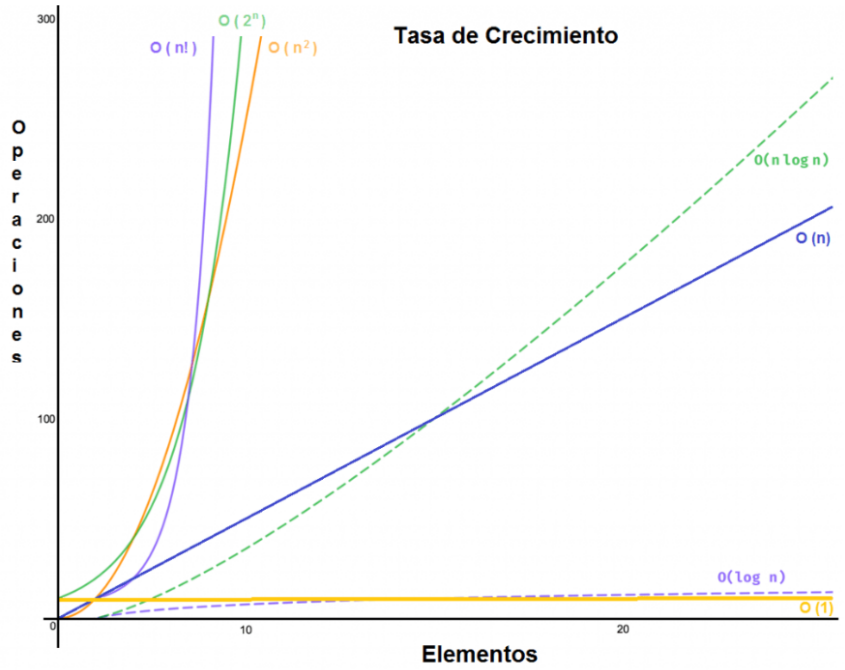
Propiedades

- Si $g \in O(f)$ y $h \in O(f)$, entonces $g + h \in O(f)$
- Si $f \in O(g)$ y $g \in O(h)$, entonces $f \in O(h)$
- $f + g \in O(max(f, g)$
- $O(f + g) = O(max(f, g))$
- Si $f \in O(f')$ y $g \in O(g')$, entonces $f + g \in O(f' + g')$
- Si $f \in O(f')$ y $g \in O(g')$, entonces $f \cdot g \in O(f' \cdot g')$
- Si $f \in O(g)$ y $n \geq 1$, entonces $f^n \in O(g^n)$

Ordenes de complejidad

Principales ordenes de complejidad

Orden	Nombre
O(1)	constante
O(log n)	logarítmica
O(n)	lineal
O(n log n)	casi lineal
O(n²)	cuadrática
O(n³)	cúbica
O(a^n)	exponencial



Jerarquía de complejidad

$$O(1) \subset O(\log n) \subset O(n)n \subset O(n \log n) \subset O(n^2) \subset O(n^3) \subset O(2^n)$$

Comparación

Efecto de duplicar el dato de entrada

T(n)	n = 100	n = 200
log(n)	1 h.	1.15 h.
n	1 h.	2 h.
nlog(n)	1 h.	2.30 h.
n^2	1 h.	4 h.
n^3	1 h.	8 h.
2^n	1 h.	$1.27*10^{30}$ h.

Efecto de duplicar el tiempo disponible

T(n)	t = 1h	t = 2h
log(n)	n = 100	n = 10000
n	n = 100	n = 200
nlog(n)	n = 100	n = 178
n^2	n = 100	n = 141
n^3	n = 100	n = 126
2^n	n = 100	n = 101

```
def codigo_1( number ):
  a = 0
  for j in range(1, number+1):
    a += a + j

  for k in range(number, 0, -1):
    a -= 1
    a *= 2
  return a
```

Determine

- Número de operaciones
- Orden de Complejidad

```
def codigo_2():
  a = 0
  a -= 1
  a *= 2
```

```
def codigo_3( number ):
  a = 0;

  for j in range(1, number+1):
    for k in range(1, number+1):
      a += a + ( k*j )

  return a
```

Determine

- Número de operaciones
- Orden de Complejidad

De una lista L ordenada de n elementos, encuentre el valor x, indique donde se encuentra el valor x.

De una lista L ordenada de n elementos, encuentre el valor x, indique donde se encuentra el valor x.

Compara x con el elemento de la mitad (mid)
if $x = mid$ **then**
    Regresa el indice de mid
else if $x > mid$ **then**
    x vive del lado derecho de mid
else
    x vive del lado izquierdo de mid
end if

- ¿cuál es su complejidad computacional?

- Problemas P: problemas de optimización para los cuales existe un algoritmo de orden $O(p(n))$ donde $p(n)$ es un polinomio.
- Problemas NP: Problemas para los cuales es posible verificar la validez de una solución en tiempo polinomial, pero no hay garantía que exista un algoritmo de orden polinomial.
- NP-Completo: Un problema que no se puede resolver en un tiempo polinomial.

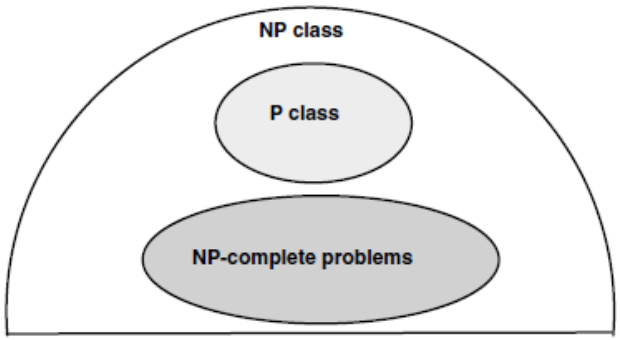


FIGURE 1.6 Complexity classes of decision problems.

	P	NP	NP-Completo
Definición	Suelen ser la clase de problemas computacionales que son tratables, es conocido por contener muchos problemas naturales.	Son los problemas que no se pueden resolver por un algoritmo, para ello es necesario utilización de técnicas o métodos heurísticos sobre los cuales el algoritmo puede determinar la proximidad de una posible solución.	Son aquellos problemas los cuáles están en la frontera externa de la clase np, son los peores problemas posibles de la clase np.
Diferencias	En P están los problemas que se pueden resolver en tiempo polinómico.	El algoritmo se utiliza sobre una técnica heurística para poder aproximarse a una solución.	Imposible encontrar un algoritmo eficiente para una solución óptima.
Ejemplos	Búsqueda binaria, secuencial, factorial	Torres de Hanói, ordenación, Shell.	Vendedor viajero, mochila, ciclo hamiltoniano, coloración.

Tarea 1

Supongamos que tenemos m almacenes que tiene que suministrar a n clientes un determinado producto. Se sabe que la capacidad de oferta de cada origen (almacén) i es a_i mientras que la demanda de cada destino (cliente) j es d_j. Además, el costo por enviar una unidad de producto del almacén i al destino j es c_{ij}. El problema consiste en determinar la cantidad de producto que deben enviarse desde el origen i al destino j de forma que se minimicen los costos de envío garantizando la demanda de los destinos sin exceder la capacidad de los orígenes oferta, o_i.

- Plantea la formulación del problema. (30 puntos)
- Argumenta sobre la complejidad del problema y el uso de heurísticos. (20 puntos)
- Elabora un algoritmo heurístico para obtener una solución factible al problema, se debe realizar lo siguiente:
  - Idea del algoritmo. Responder ¿Qué hace? ¿Cómo se hace?
  - Pseudocódigo o diagrama de flujo.
  - ¿Cuántas operaciones realiza su algoritmo propuesto?
  - ¿Cuál es la complejidad de su algoritmo propuesto?

Tarea 2

A cada equipo se le asignará un problema clásico de optimización y elaborará una infografía. Debe contener las siguientes secciones: Definición del problema, variables de decisión, limitantes, objetivo, formulación matemática y complejidad computacional. Durante la clase siguiente, cada equipo expondrá la infografía (5 min).

Equipo	Problema
1	The Assigment problema
2	Traveling salesman problem
3	The set covering problema
4	Uncapacitated facility location

Equipo	Problema
5	Uncapacitated lot sizing
6	Vehicle routing problem
7	Set partitioning problem
8	The transportation problem