Problemas de alta complejidad y Técnicas de resolución

Complejidad de un algoritmo

- Todo algoritmo tiene una serie de características, entre otras que requiere de recursos, algo que es fundamental considerar a la hora de implementarlos en una maquina. Estos recursos son principalmente:

  – El tiempo: período transcurrido entre el inicio y la finalización del algoritmo.
  – La memoria: la cantidad (la medida varía según la máquina) que necesita el algoritmo para su ejecución.

Complejidad de un algoritmo

- Los modelos de complejidad son normalmente usados para
  - Mostrar que un algoritmo dado es optimo o
  - Determinar la complejidad mínima de un problema (establecer el limite superior del numero de operaciones necesarias para resolver un problema dado).
- **Complejidad Computacional:** considera globalmente todos los posibles algoritmos para resolver un problema planteado

Problemas NP-completos,

- P es la clase de problemas de decisión que se pueden resolver mediante un algoritmo de tiempo polinomico.
- Un algoritmo en tiempo polinómico es eficiente si existe un polinomio p(n) tal que el algoritmo puede resolver cualquier caso de tamaño n en un tiempo O(p(n))

- ¿Qué es un algoritmo eficiente? O(nlgn), $O(n^2)$?
  - Algoritmo eficiente: ∃ p(n) | él puede resolver cualquier caso de tamaño *n en tiempo O(p(n))*

    ⇒ Algoritmo de tiempo polinómico.

Problemas NP-completos,

- Teoría de la NP-Completitud:
  - Problemas sin algoritmo eficiente, de dificultad intrínseca que en algunos casos aún no ha sido demostrada.
  - Se cree que algoritmos eficientes para ellos no existen. Lo que sí es eficiente es la validación de una supuesta solución.

.

Problemas de alta complejidad: optimización

Qué es un problema de optimización combinatoria?

Dado un conjunto finito N = {1,….n} con pesos c_j asignados a cada $j \in N$ y *F* una familia de conjuntos factibles de N, entonces queremos encontrar el elemento de F de peso mínimo:

$$\text{Min}_S \{ \sum_{j \in S} c_j / S \in F \}$$

(también podría ser un problema de maximización).

Problemas de alta complejidad: optimización

- Problemas de optimización combinatoria: maximizan o minimizan funciones de varias variables sujeto a restricciones de integridad sobre todas o algunas variables.

- Ejemplos de aplicación: distribución de productos o mercaderías, secuenciamiento de tareas en problemas de producción, diseño de redes de comunicación, etc.

$$\text{Max } cx + hy \,/\, Ax + Gy \leq b, \text{ con } x \in Z^n_+, y \in R^n_+$$

Problemas de alta complejidad: optimización

- Def: *región factible* S:

  $S=\{(x,y) / x \in Z^n_+, y \in R^n_+, Ax+ Gy \leq b\}$

- Def: *solución óptima*: una solución factible (x_0,y_0) es *óptima* si $\forall (x,y) \in S$, $cx_0+hy_0 \geq cx +hy$

- Def: *valor óptimo* = cx_0+hy_0

Problemas de alta complejidad: optimización

- No hay definición unánime de *optimización combinatoria*.

- <u>Def</u> Sea N={1..n} finito y c=(c_1,...c_n), para F $\subseteq$ N, definimos c(F)=Σc_j, y $\Gamma = 2^N$, un problema de *optimización combinatoria* (cp) es :

$$(cp)\ \text{Max } c(F) / F \in \Gamma$$

- En optimización combinatoria se busca la solución en un conjunto finito o en un contable infinito

Problemas de optimización combinatoria

maximizan o minimizan funciones de varias variables sujeto a restricciones de integridad sobre todas o algunas variables.

Un pastor tiene que pasar un lobo, un conejo y una col de una orilla de un río a la otra orilla. Dispone de una barca en la que sólo caben él y una de las tres cosas anteriores. Si deja solos al conejo y al lobo, éste se come a aquél; si deja al conejo con la col, aquél se la come.
¿Cómo debe proceder para llevar las tres cosas a la orilla opuesta?

Problemas de optimización combinatoria

Mínimo global vs mínimo local (fuerte o débil)

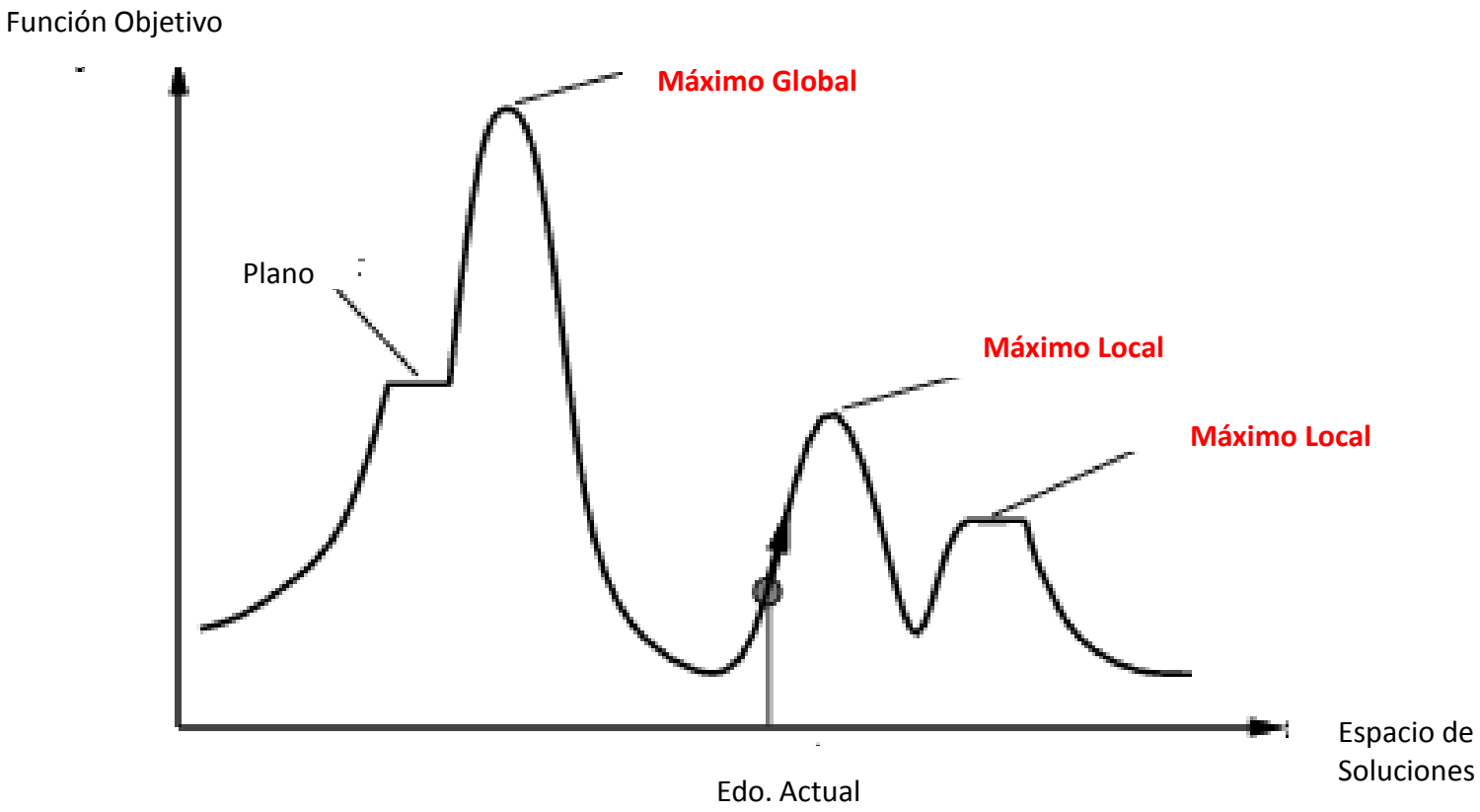
Con restricciones y sin restricciones

Multiobjetivos vs. un único objetivo

Múltiples soluciones vs. una única solución

Dinámico vs. estático

Problemas de optimización combinatoria



Problemas NP-completos,

- TSP
- HAM
- Mochila
- Ciclos eulerianos
- Buscaminas, Tetris
- Factorización
- Cobertura de vértices

Ejemplos Problemas de alta complejidad: optimización:

- **Problema de asignación:** Tenemos n personas para realizar n trabajos. Cada persona hace un sólo trabajo. Se asigna un costo c_{ij} a la asignación de la persona i para hacer el trabajo j de acuerdo a la capacidad de cada uno para realizar cada trabajo. Se quiere encontrar la asignación que minimice los costos.

- **Problema de la mochila (Knapsack problem)** Se tiene un presupuesto de b pesos disponible para invertir en algunos de n posibles proyectos para el próximo año. Sea a_j la inversión que requiere el proyecto j y c_j el beneficio que produce dicho proyecto. Se quiere maximizar las ganancias (sólo se puede invertir en el proyecto completo).

Ejemplos Problemas de alta complejidad: optimización:

Problema del viajante de comercio (Traveling Salesman Problem, TSP): (es el problema más famoso de Optimización Combinatoria y el más estudiado.) Un viajante de comercio quiere visitar n ciudades y volver al lugar de partida en el menor tiempo posible (o al menor costo).

Los problemas que acabamos de ver hay que buscar el subconjunto óptimo entre una familia de subconjuntos. Uno podría pensar en enumerar todos estos conjuntos y elegir el mejor (algoritmo de fuerza bruta).

Sirve este procedimiento?.

Por ejemplo en el caso del viajante de comercio, si suponemos que podemos ir de cualquier ciudad a cualquier otra, tendríamos que revisar (n-1)! recorridos posibles. Por ejemplo para un problema con n= 101 ciudades hay 9.33×10^{157} circuitos (recorridos) posibles que habría que revisar.

Problemas de alta complejidad: Explosión combinatoria

¿Cuál es el número de posibles ordenaciones de una baraja de póker de 52 cartas?

El resultado es *52!*, que es aproximadamente 8×10^{67}. Observa que a partir de una simple baraja obtenemos un enorme número, superior, por ejemplo, al cuadrado del número de Avogadro: $6{,}02 \times 10^{23}$.

Problemas de alta complejidad: optimización

- **0-1 Knapsack**

  Instancia:

  - **Sean n alimentos/ costo a_j y una capacidad nutritiva c_j $\forall$ j=1..n . Cada alimento puede ser elegido para ser llevado o no y no se puede fraccionar.**
  - **Existe un presupuesto b o capacidad de la mochila que no puede ser superado.**
  - **Pb: elegir un subconjunto de alimentos que maximice el valor nutritivo no excediendo el presupuesto (capacidad de la mochila)**

$$\text{Max } \Sigma\, c_j x_j \,/\, \Sigma\, a_j x_j \leq b \text{ , } x=(x_1,..x_n) \in B^n$$

Problemas de alta complejidad: optimización

- **Asignación**
  - **Sean n personas y m trabajos con $n \geq m$.**
  - **Cada trabajo debe ser hecho por una persona y cada persona puede hacer a lo sumo un trabajo**
  - **c_{ij}=costo de una persona j haciendo un trabajo i**
  - **Pb: encontrar una asignación de personas a trabajos /minimice el costo total de completar todos los trabajos:**

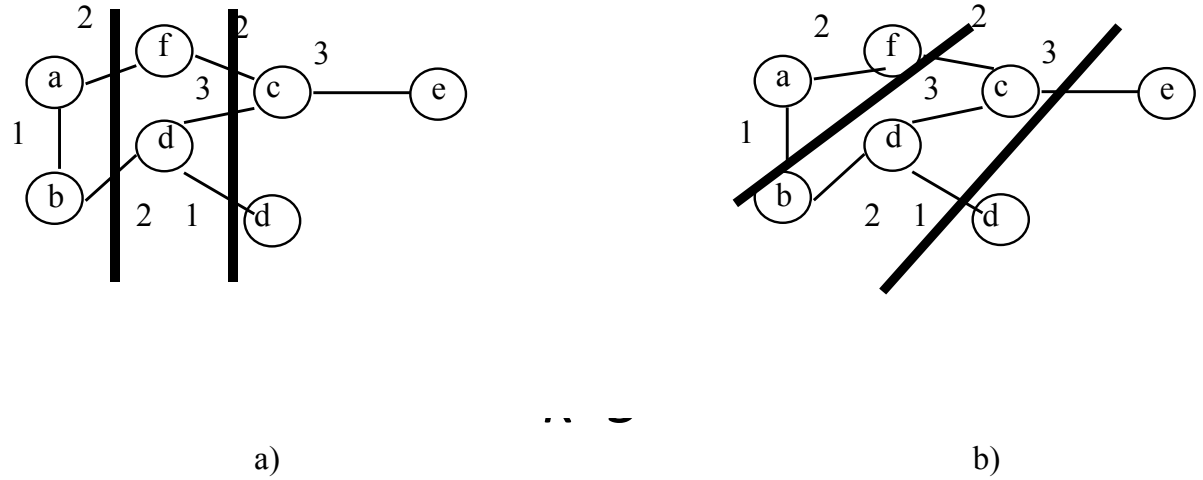
    $x_{ij} \in B=\{0,1\}$, $x_{ij}=1$ si persona j asignada a trabajo i, si no $x_{ij}=0$

    $\sum_j x_{ij} = 1$, $\sum_i x_{ij} \leq 1$

  - **Función objetivo: Min $\sum \sum c_{ij} x_{ij}$**
  - **Existen m+n elementos y se particionan en 2 conjuntos disjuntos de trabajos y personas.**

Problema de k-partición de grafos

- Objetivo: asignar los vértices tal que se minimice los arcos en diferentes sub-grafos y se repartan los vértices equitativamente entre los diferentes sub-grafos.



a) b)

Problema de partición de grafos

- Definición de una Función de Costo General
  - *Costo de Comunicación*
  - *Costo por el Desequilibrio de la Carga*

$$F = A_1 C_C + A_2 C_D$$

Objetivo: MIN(*F*)

Problemas de alta complejidad: optimización

EJEMPLOS

- **Traveling Salesman Problem (TSP):**

  Instancia:

  - **dado un entero n>0 una matriz de distancias $[d_{ij}]_{nxn}$ entre cualquier par de n ciudades / $d_{ij} \in Z_+$.**
  - **tour: es un camino cerrado /visita cada ciudad exactamente una vez**
  - **Pb: encontrar un tour con largo total mínimo, o sea: F= {permutaciones de n objetos} y c: F$\rightarrow$ R asigna a cada $\pi \in$ F, un costo $\Sigma\, d^{\pi}_{ij}$**

El problema del Vendedor.

- Un vendedor quiere vender su producto en varias ciudades.
- Las ciudades están representadas por vértices.
- El coste para ir de una ciudad a otra por las aristas.
- Se puede suponer que el grafo es completo.

El *Agente Viajero*

Problema de optimización combinatoria

- *Formalización:* Dada N ciudades, el viajero de comercio debe visitar cada ciudad una vez, teniendo en cuenta que el costo total del recorrido debe ser mínimo.

G=(N, A)

N={1, ...,n}: conjunto de *n* nodos (vértices)

A={a_{ij}}: matriz de adyacencia.

d_{ij} = { ∞ Si $a_{ij} = 0$

Lij Si $a_{ij} = 1$

Lij: distancia entre las ciudades i y j.

El *Agente Viajero*

- Suponiendo que las ciudades son numeradas desde 1 hasta n, una solución al problema puede expresarse a través de una matriz de estado *E*

e_{ij} = { 1 Si la ciudad *j* fue la i^{esima} ciudad visitada

0 En otro caso

- La matriz E permite definir un arreglo unidimensional V de dimensión *n*;

$v_j = i$ Si la ciudad i fue la j^{esima} ciudad visitada

El *Agente Viajero*

- **Función objetivo**

$$F1 = \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n} L_{ik} e_{ij} e_{kj+1}$$

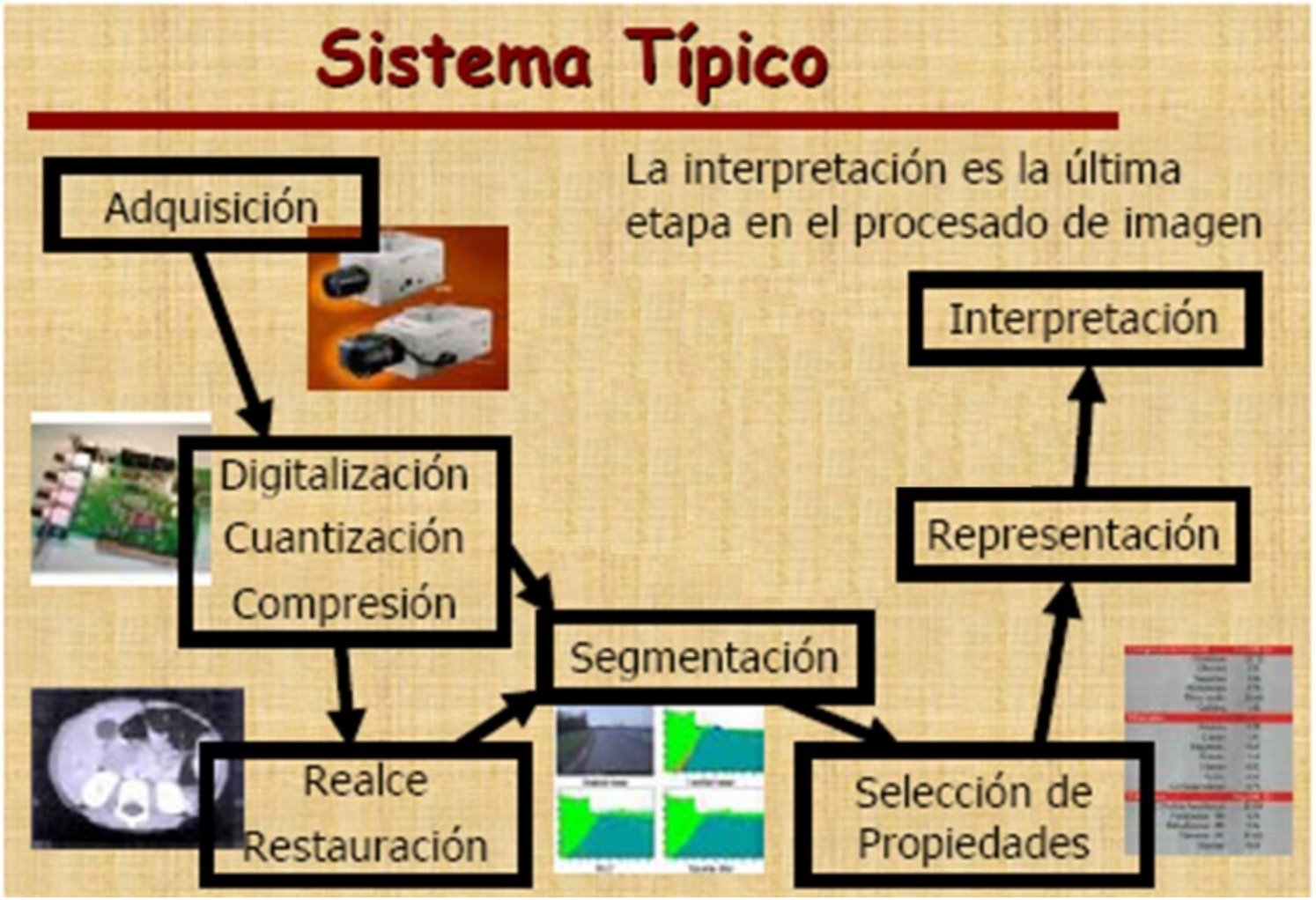
$$F2 = C\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\left(e_{ij} - n\right) + \sum_{i=1}^{n}\sum_{j=1}^{n}\left(e_{ij} - 1\right) + \sum_{j=1}^{n}\sum_{i=1}^{n}\left(e_{ij} - 1\right)\right)$$

FC = F1 + F2

C = n * Max(L_{ik}) factor de penalización.

Problemas de alta complejidad: reconocimiento

Reconocimiento



Reconocimiento

un **sistema de reconocimiento** de patrones consiste en:

• Un **sensor** que recoge las observaciones a clasificar.

• Un **sistema de extracción de características** transforma la información observada en valores numéricos o simbólicos.

• Un **sistema de clasificación** o descripción que, basado en las características extraídas, clasifica la medición.

Reconocimiento

El punto esencial del reconocimiento de patrones es la **clasificación.**

Se quiere clasificar una imagen dependiendo de sus **características**.

Por ejemplo se puede clasificar imágenes digitales de letras en las clases «A» a «Z» dependiente de sus píxeles o se pueden clasificar huellas dactilares.

Reconocimiento

PATRÓN: CONJUNTO DE CARACTERÍSTICAS DE UN OBJETO

Por ejemplo, las características de una imagen pueden ser:

-topológicas: número de componentes conexas, agujeros,…

-geométricas: área, perímetro, curvatura,…

-estadísticas: momentos,…

Una **clase de patrones** es un conjunto de patrones **similares.**

El objetivo del reconocimiento de patrones es **asignar un patrón a la clase a la que pertenece** (lo más automáticamente posible).

Reconocimiento

PATRÓN: CONJUNTO DE CARACTERÍSTICAS DE UNA IMAGEN

Reconocimiento

Ejemplo: supongamos que queremos discriminar tres tipos de flores (virginica, versicolor, setosa) mediante las características:

- la anchura de sus pétalos.
- la longitud de sus pétalos.

En este caso, un patrón consta de estas dos medidas tomadas de una flor en particular.

Una clase de patrones consistiría en el conjunto de todos los patrones que se obtienen de la misma clase de flor.

Reconocimiento

Podemos almacenar los patrones en diversos formatos:

- El más usual es el de **vector** (para características cuantitativas)

$$\boldsymbol{x}=[x_1, x_2, \ldots, x_n]^T$$

donde $\boldsymbol{x}$ es el patrón y x_i son las características

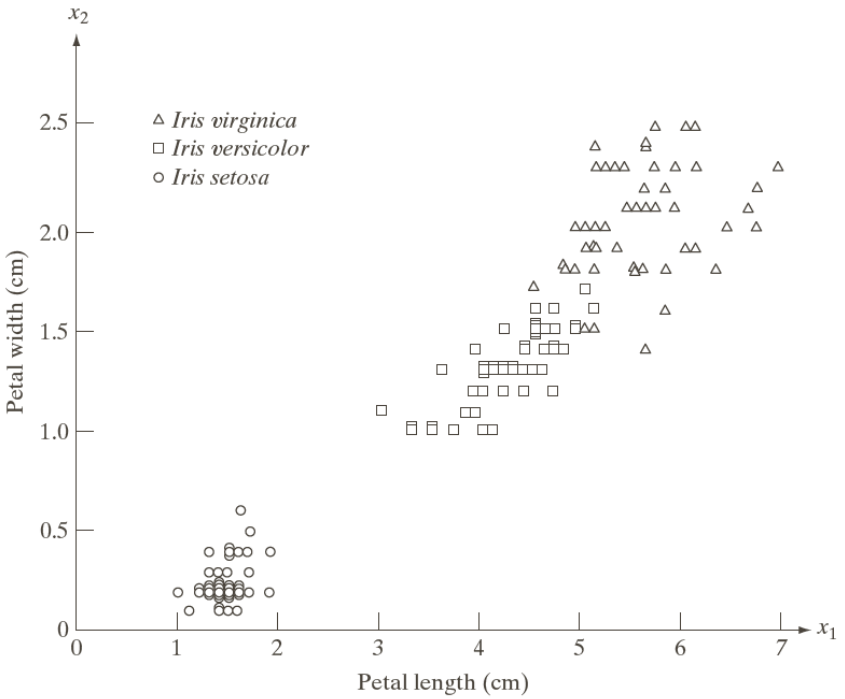
- También se usa el formato de árbol (para características estructurales).

Reconocimiento

Ejemplo: supongamos que queremos discriminar tres tipos de flores (virginica, versicolor, setosa) mediante las características:

- la anchura de sus pétalos (x_1)
- longitud de sus pétalos (x_2).

En este caso, el patrón es $\boldsymbol{x}=[x_1, x_2]^T$



Como podemos observar, esta elección de características podrá discriminar perfectamente la clase setosa de las otras dos, pero no así las clases virginica y versicolor entre sí.

FIGURE 12.1
Three types of iris flowers described by two measurements.

Reconocimiento

Ejemplo: supongamos que queremos reconocer imágenes de satélites.

FIGURE 12.4
Satellite image of a heavily built downtown area (Washington, D.C.) and surrounding residential areas. (Courtesy of NASA.)

En este caso, usaríamos una estructura en árbol:

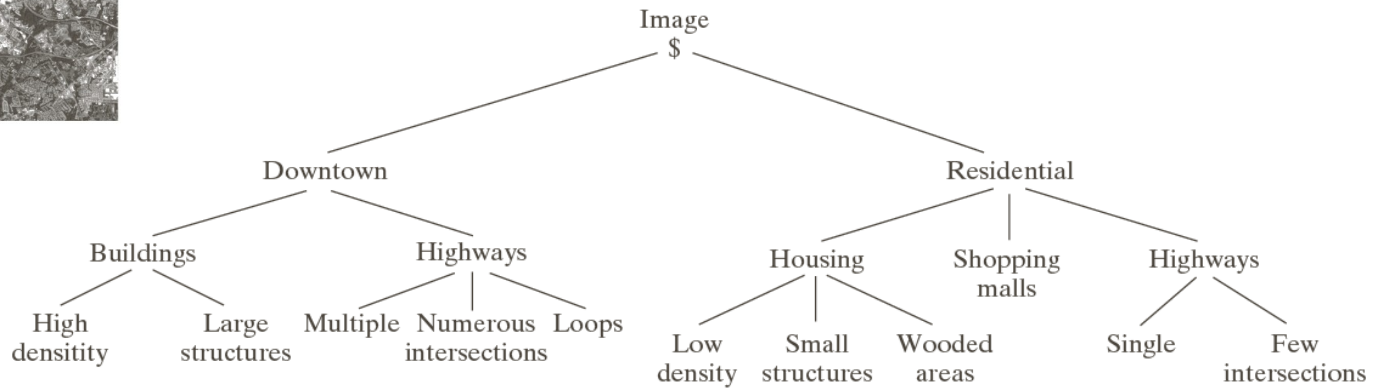


FIGURE 12.5 A tree description of the image in Fig. 12.4.

Reconocimiento

La **clasificación** utiliza habitualmente uno de las siguientes procedimientos:

• **clasificación estadística** (o teoría de la decisión), basado en las características estadísticas de los patrones.

• **clasificación sintáctica** (o estructural), basado en las relaciones estructurales de las características.

La clasificación puede ser de dos tipos:

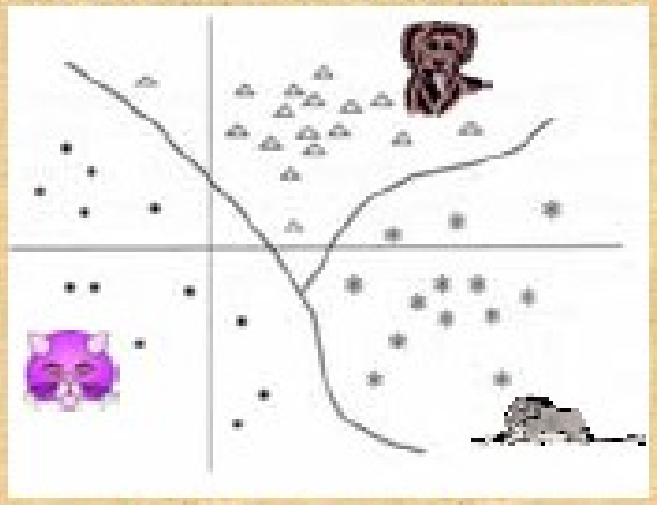
• **Supervisada**, si se usa un conjunto de aprendizaje, que sirve para entrenar al sistema.

• **No supervisada**. El sistema no tiene un conjunto para aprender a clasificar la información a priori, sino que se basa en cálculos estadísticos para clasificar los patrones.

Reconocimiento

Clases Separables

Si existe una **hipersuperficie de discriminación** que separa el espacio de propiedades en regiones que contienen sólo un tipo de objetos, **las clases serán separables**



En la mayor parte de los problemas de reconocimiento en Tratamiento Computacional de Imagen las clases no son separables

Reconocimiento

RECONOCIMIENTO BASADO EN MÉTODOS DE DECISIÓN

Sea $\boldsymbol{x}=[x_1, x_2, \ldots, x_n]^T$ un patrón donde cada x_i es una característica.

Para cada clase **w** de patrones, hay que encontrar una función de decisión d_w **(clasficador)** con la propiedad de que si ***x*** pertenece a la clase ***w*** y no a la clase **v** , entonces

$$d_w(x) > d_v(x)$$

Frontera de decisión: aquellos vectores **x** tales que $d_w(x) = d_v(x)$. Considerando la función frontera $d_w(x) - d_v(x)$, dicha función tomará valores >0 cuando **x** pertenezca a la clase **w** y valores <0 cuando pertenezca a **v**.

Reconocimiento

RECONOCIMIENTO BASADO EN MÉTODOS DE DECISIÓN

Clasificador por la mínima distancia:

Sea $\boldsymbol{x}=[x_1, x_2, \ldots, x_n]^T$ un patrón donde cada x_i es una característica. El clasificador más sencillo es el de la mínima distancia hacia el vector promedio de todos los patrones que pertenecen a la clase:

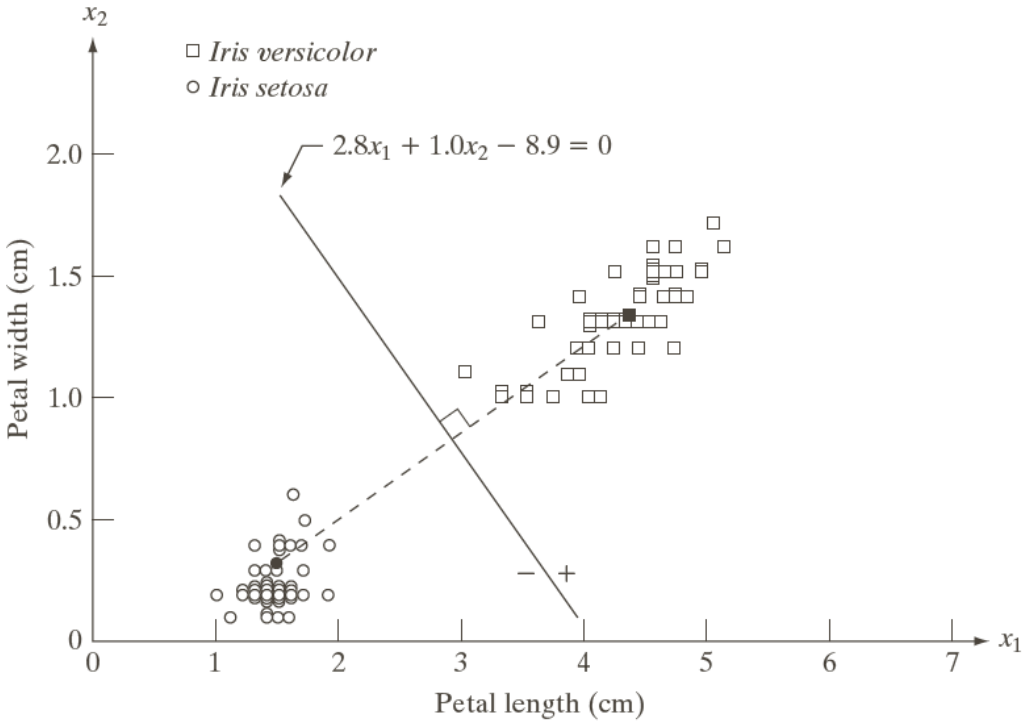


FIGURE 12.6 Decision boundary of minimum distance classifier for the classes of *Iris versicolor* and *Iris setosa*. The dark dot and square are the means.

Reconocimiento

RECONOCIMIENTO BASADO EN MÉTODOS DE DECISIÓN

Clasificador por la mínima distancia:

Fase de aprendizaje: Una vez aislada las propiedades características de cada clase, ésta se representa por un patrón de clase que consiste en la media de todos los patrones que pertenecen a la misma clase:

$$m_j = 1/N_j \sum_{x \epsilon \omega_j} x \text{ for } j = 1, 2, ..., M$$

siendo N_j el número de patrones correspondientes a la clase w_j

Fase de clasificación: para cada vector de características *x*, determinamos su proximidad a cada patrón de clase $\boldsymbol{m_j}$. Si escogemos la distancia euclídea para determinar la proximidad, tenemos que calcular:

$$D_j(x) = \|x - m_j\| \text{ for } j = 1, 2, ..., M$$

Ahora le asignamos a *x* la clase *de* m_j si $D_j(x)$ es el mínimo de todas las distancias. Esto es equivalente a imponer que la función de decisión

$$d_j(x) = x^T m_j - 1/2(m_j^T m_j) \text{ for } j = 1, 2, ..., M$$

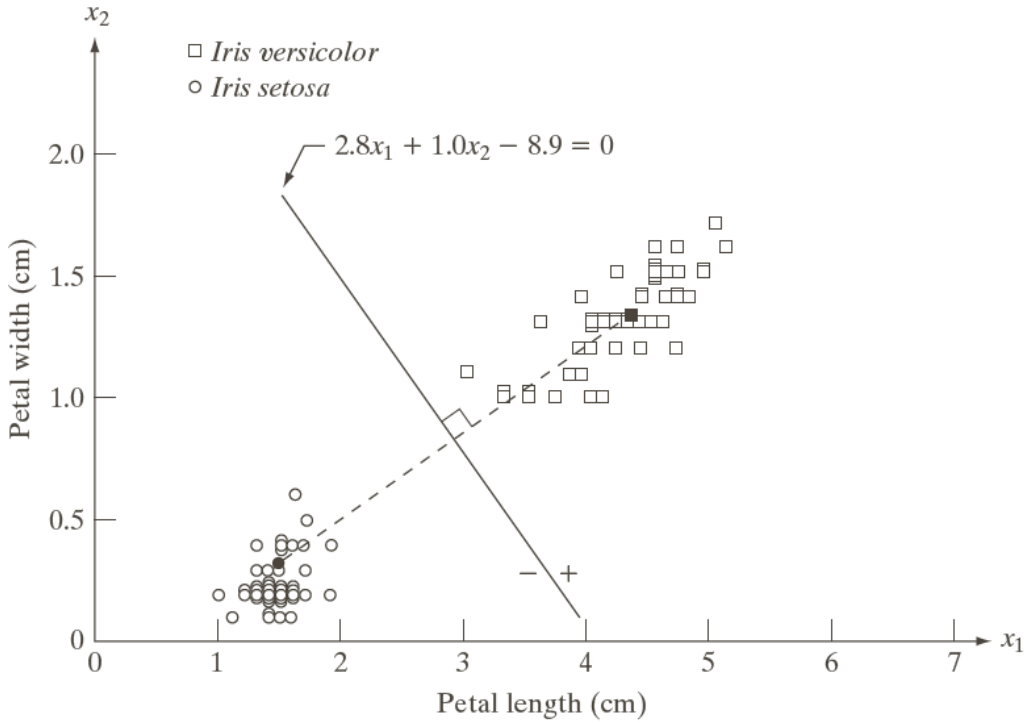
tome un valor máximo.

Reconocimiento

RECONOCIMIENTO BASADO EN MÉTODOS DE DECISIÓN

Clasificador por la mínima distancia:

Ejemplo



$$m_j = 1/N_j \sum_{x \in \omega_j} x$$

$$m_1 = (4.3,1.2); m_2 = (1.5,0.3)$$

$$d_j(x) = x^T m_j - 1/2(m_j^T m_j)$$

$$d_1(x) = 4.3x_1 + 1.3x_2 - 10.1$$

$$d_2(x) = 1.5x_1 + 0.3x_2 - 1.17$$

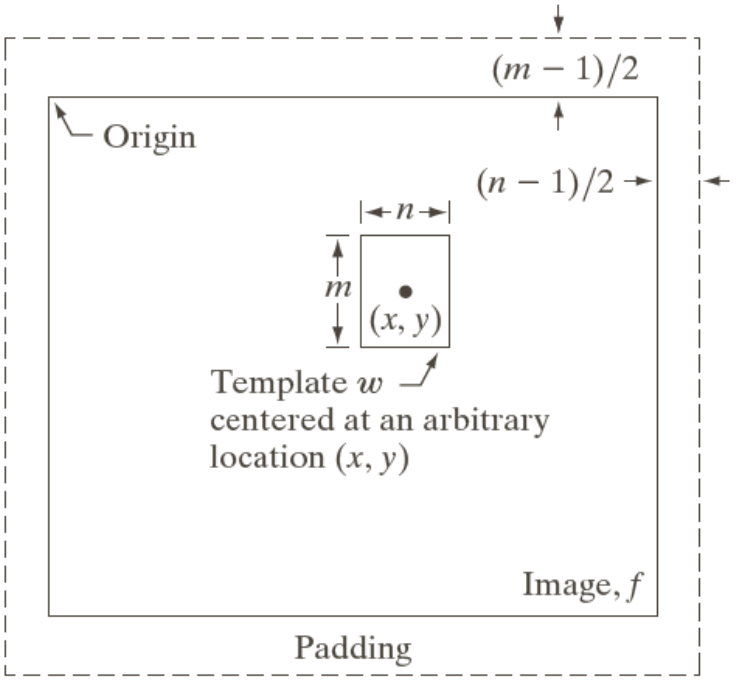
Función frontera

$$d_{1,2}(x) = 2.8x_1 + 1.0x_2 - 8.9 = 0$$

Reconocimiento

RECONOCIMIENTO BASADO EN MÉTODOS DE DECISIÓN

Clasificador por correlación:



•Se trata de calcular un coeficiente de correlación normalizado mediante la correlación espacial con una “muestra” del objeto que queremos reconocer dentro de la imagen. Este coeficiente está entre -1 y 1 y toma el máximo valor (1) cuando se da una total coincidencia.

$$\gamma(x,y)=\frac{\sum_{s,t}(w(s,t)-\overline{w})\sum_{s,t}(f(x+s,y+t)-\bar{f}_{s,t})}{\sqrt{\sum_{s,t}(w(s,t)-\overline{w})^2\sum_{s,t}(f(x+s,y+t)-\bar{f}_{s,t})^2}}$$

Donde $\overline{w}$ es la media en la muestra *w, y* $\bar{f}_{s,t}$ es el valor medio de *f* en la región coincidente con *w.*

Reconocimiento

RECONOCIMIENTO BASADO EN MÉTODOS DE DECISIÓN

Clasificador por correlación:

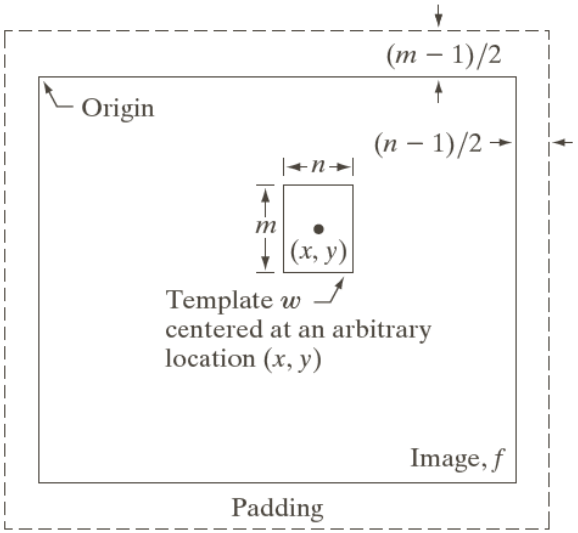
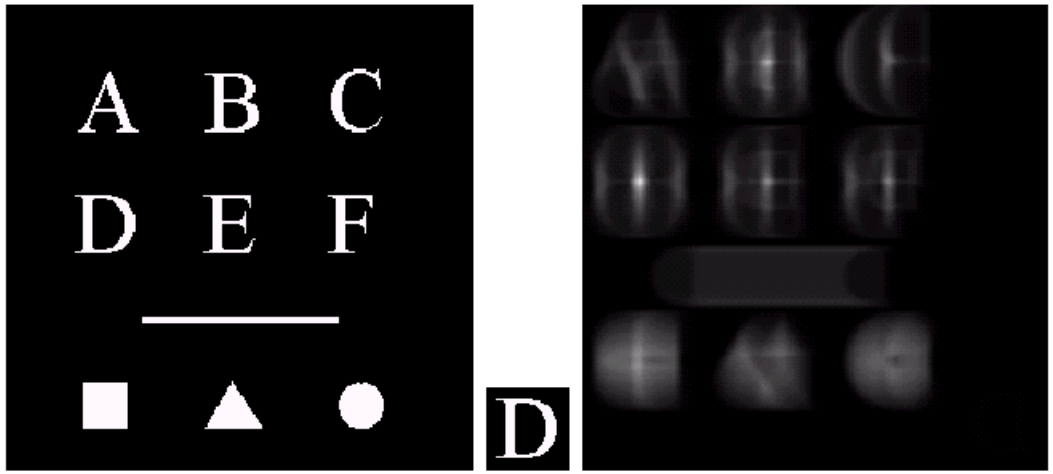


FIGURE 12.8
The mechanics of template matching.

Ejemplo:



a b c

FIGURE 12.9
(a) Image.
(b) Subimage.
(c) Correlation coefficient of (a) and (b). Note that the highest (brighter) point in (c) occurs when subimage (b) is coincident with the letter "D" in (a).

Reconocimiento

RECONOCIMIENTO BASADO EN MÉTODOS DE DECISIÓN

Otros ejemplos de clasificadores:

- Clasificadores mediante árboles de decisión
- Algoritmos genéticos
- Clasificador de Bayes.
- Clasificador por redes neuronales.

Reconocimiento

Problemas actuales:

- Aparatos para la captación de imágenes 3D en la vida real son todavía muy costosos y poco fiables

- El tiempo y los recursos consumidos llegan a ser enormes cuando se trata de comparar digitalmente imágenes en 3D

Problemas de alta complejidad: identificación

- Dada una colección de ejemplos de *f*, regresa una función *h* que lo aproxime

- La mas simple forma: aprender una función desde ejemplos

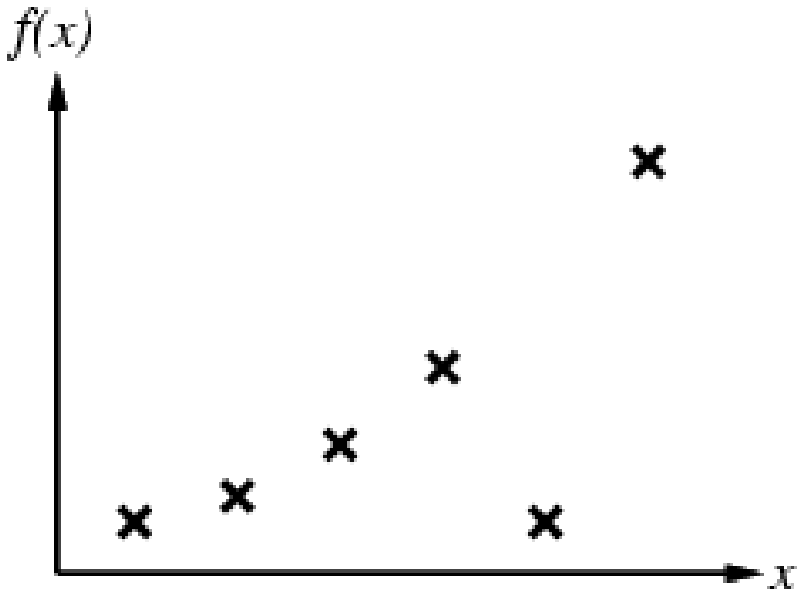
f es la función objetivo Un ejemplo es (*x*, *f(x)*)

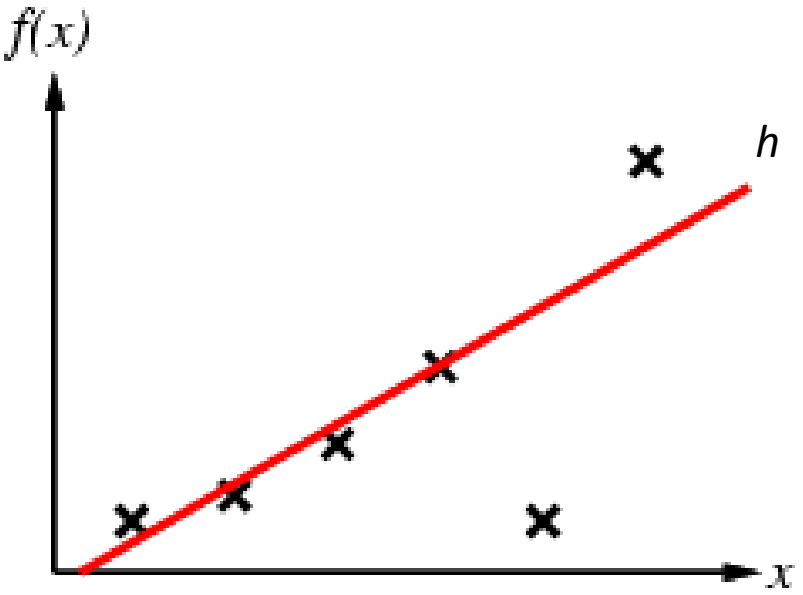
- Problema: encontrar una función hipotética *h*
  Tal que $h \approx f$
  Dado un conjunto de ejemplos

- Hipótesis
  - Ignora conocimiento a prior (en algunos casos)
  - Asume que los ejemplos son dados

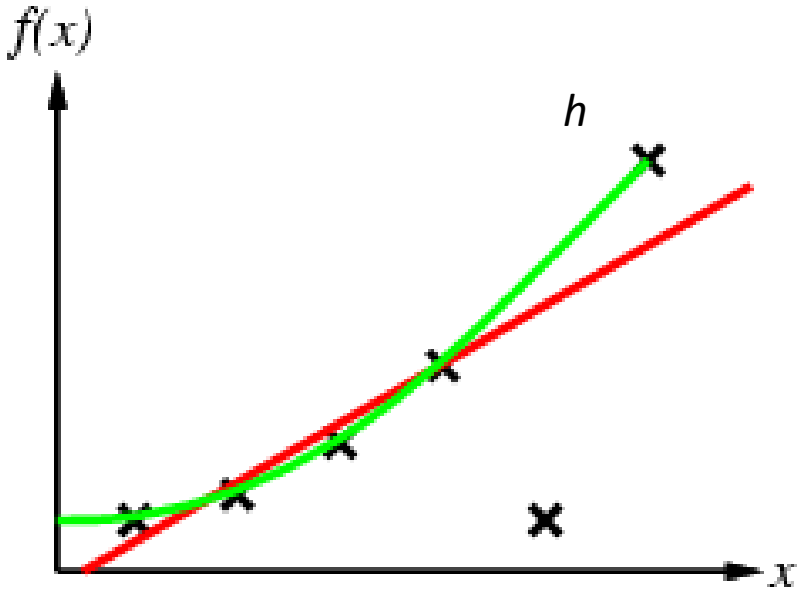
Problemas de alta complejidad: identificación

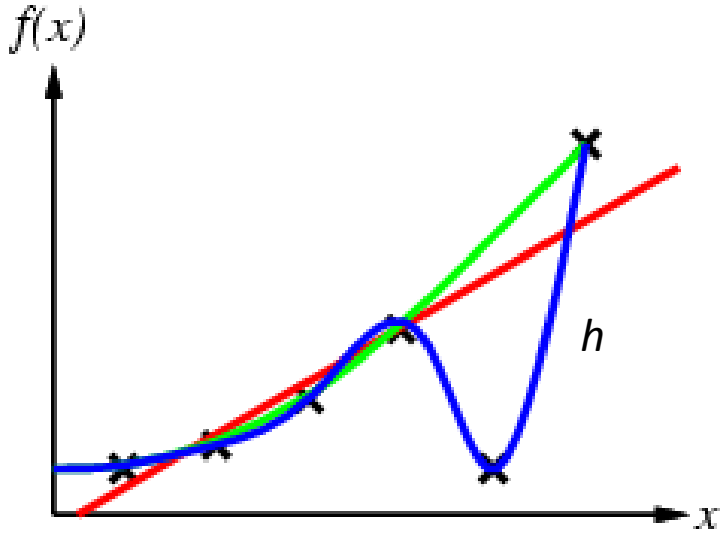
Construir *h* tal que se acerque a *f* a través de un conjunto de entrenamientos





Problemas de alta complejidad: identificación





Técnicas de resolución

- Muchos problemas de optimización combinatoria como por ejemplo el TSP se consideran problemas abiertos o no resueltos desde el punto de vista de la investigación

- No se conocen métodos buenos que resuelvan estos problemas en un tiempo razonable, o sea sólo se conocen algoritmos que requieren un número de operaciones que crece en forma exponencial cuando aumenta el tamaño del problema.

Técnicas de resolución

- Algoritmo de búsqueda:
  - Permitir la transición entre estados usando los operadores
  - Controlar esos movimientos
- Tareas de Búsqueda
  - **búsqueda ciega (Búsquedas sin contar con información)**: no utiliza información sobre el problema. No existe información acerca de los pasos necesarios o costos para pasar de un estado a otro. Normalmente, se realiza una búsqueda exhautiva.
  - **Búsqueda heuristicas (Búsqueda respaldada con información)**: usan información sobre el problema como costos, etc. se posee información muy valiosa para orientar la búsqueda para que sea mas óptima

Técnicas de resolución

Búsqueda No Informada (Ciega)

1. Búsqueda por amplitud
2. Búsqueda de costo uniforme
3. Búsqueda por profundidad
4. Búsqueda limitada por profundidad
5. Búsqueda por profundización iterativa
6. Búsqueda bidireccional

Búsqueda Informada (Heurística))

1. Búsqueda avara
2. Búsqueda A*
3. Búsqueda A*PI
4. Búsqueda A*SRM

Criterios de rendimiento

Las estrategias de búsqueda se evalúan según los siguientes criterios:

Completitud: si garantiza o no encontrar la solución si es que existe.

¿La estrategia garantiza encontrar una solución, si ésta existe?

- **Complejidad temporal:** cantidad de tiempo necesario para encontrar la solución.

¿Cuánto tiempo se necesitará para encontrar una solución?

- **Complejidad espacial:** cantidad de memoria necesaria para encontrar la solución.

¿Cuánta memoria se necesita para efectuar la búsqueda?

- **Optimalidad: si** se encontrará o no la mejor solución en caso de que existan varias.

¿Con esta estrategia se encontrará una solución de la más alta calidad, si hay varias soluciones?

Técnicas de resolución
Métodos exactos

- Branch and bound ("ramificación y poda")

Tenemos $z = \max \{ cx / x \in X \subset Z^n \}$

Queremos dividir el problema en problemas más pequeños que sean más fáciles de resolver.

O sea poner $X = X_1 \cup \ldots\ldots \cup X_h$, entonces si

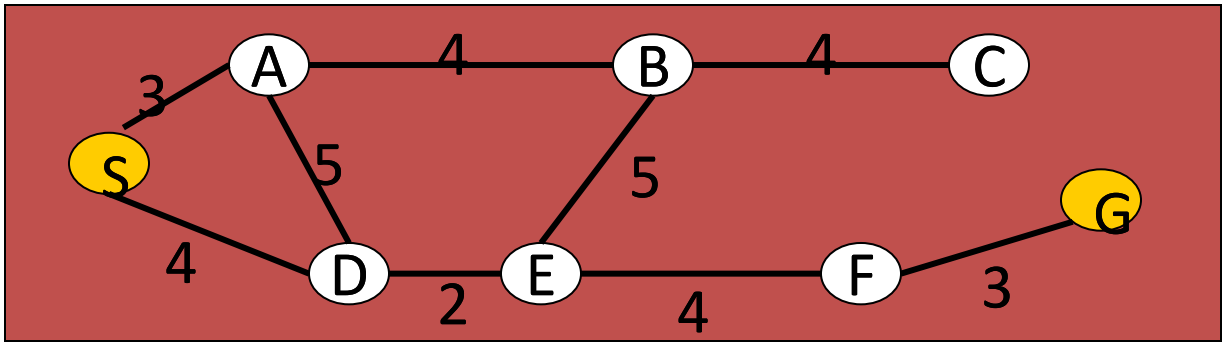
$$z^k = \max \{ cx / x \in X_k \}$$

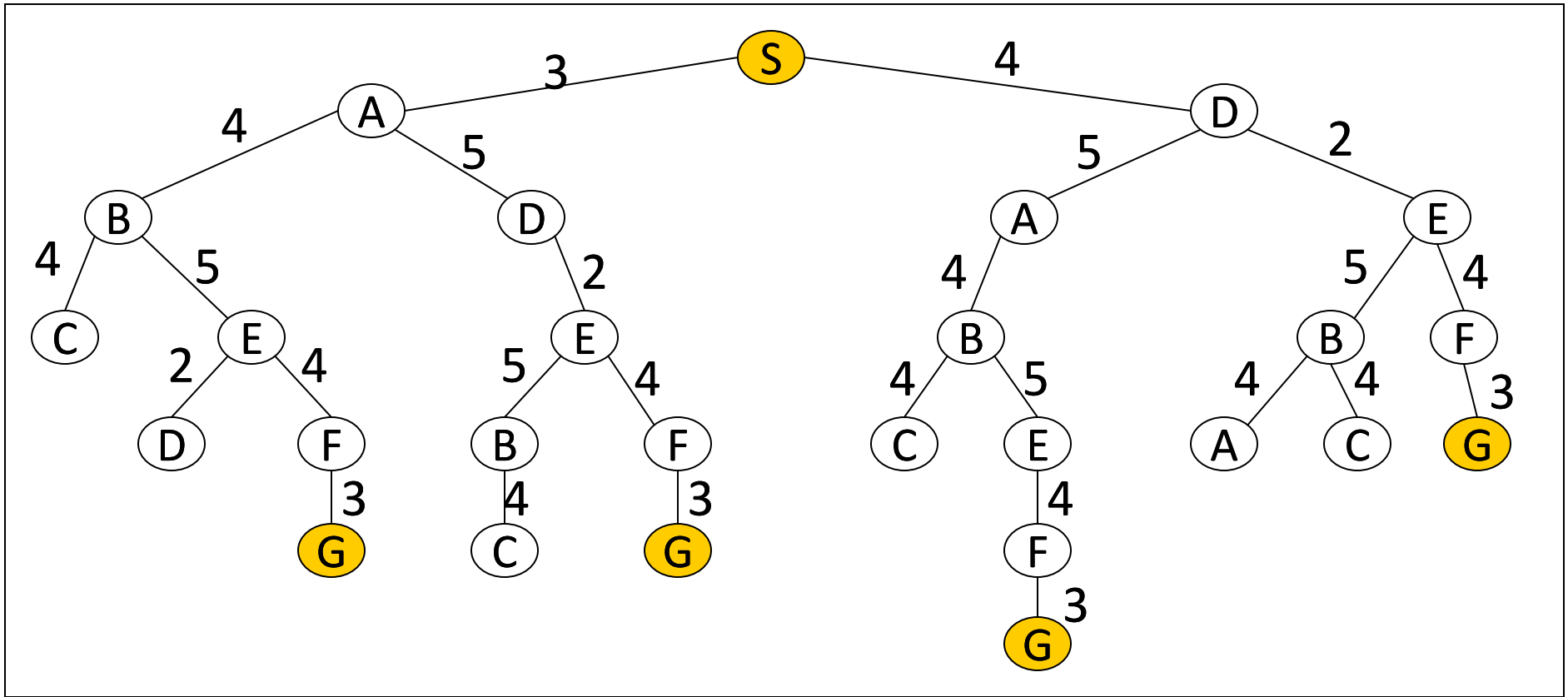
tendremos que $z = \max_k \{z^k\}$

Para construir un método de branch and bound usamos estas ideas en forma iterativa.

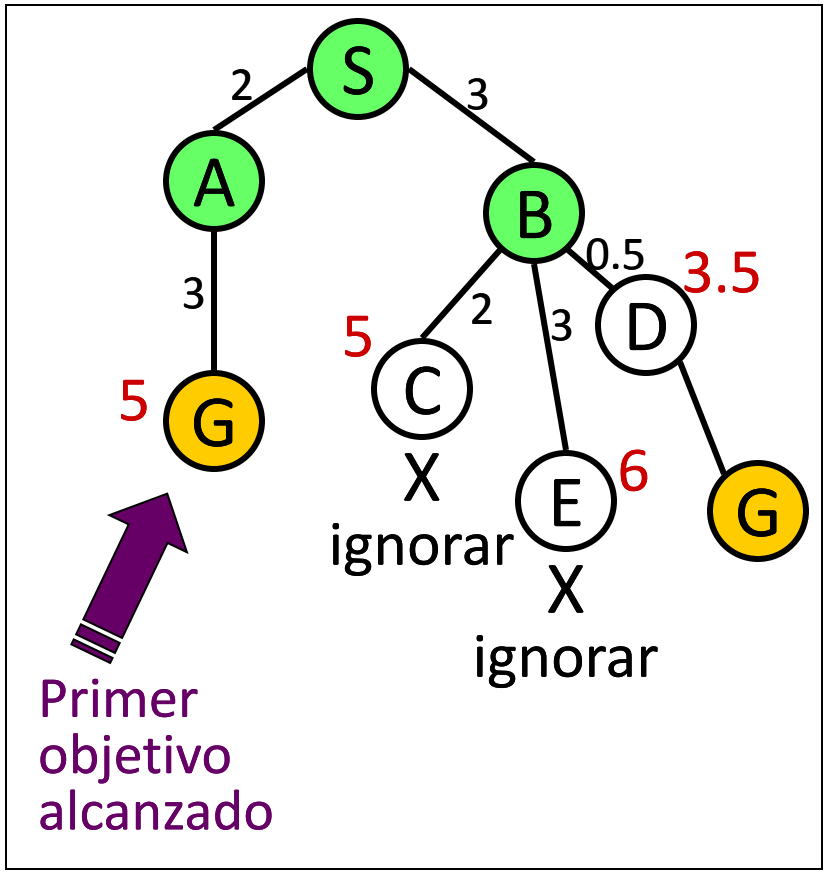
Usamos un árbol de enumeración para representar el algoritmo branch and bound. En cada nodo resolvemos un problema menor de optimización.

Técnicas de resolución





El principio Branch-and-Bound



- Usar cualquier método de búsqueda (completo) para encontrar un camino.
- Remover todos los caminos parciales que tengan un costo acumulado mayor o igual que el camino hallado.
- Continuar la búsqueda para el próximo camino.
- Iterar.

Técnicas de resolución

- Usan funciones de evaluación y se detienen al conseguir una *buena* solución
- Algoritmos heurísticas
  - Minimizar costo estimado para alcanzar objetivo desde donde estoy
  - Ejemplo Algoritmo de Búsqueda: Primero el Mejor para expandir
- Clasificación
  - Iterativo (Descenso de Gradiente, Algoritmos Genéticos, Temple Simulado) vs. Constructivo (Primero el Mejor)
  - Búsqueda Local (Temple Simulado) vs. Búsqueda Global (Algoritmos Genéticos)

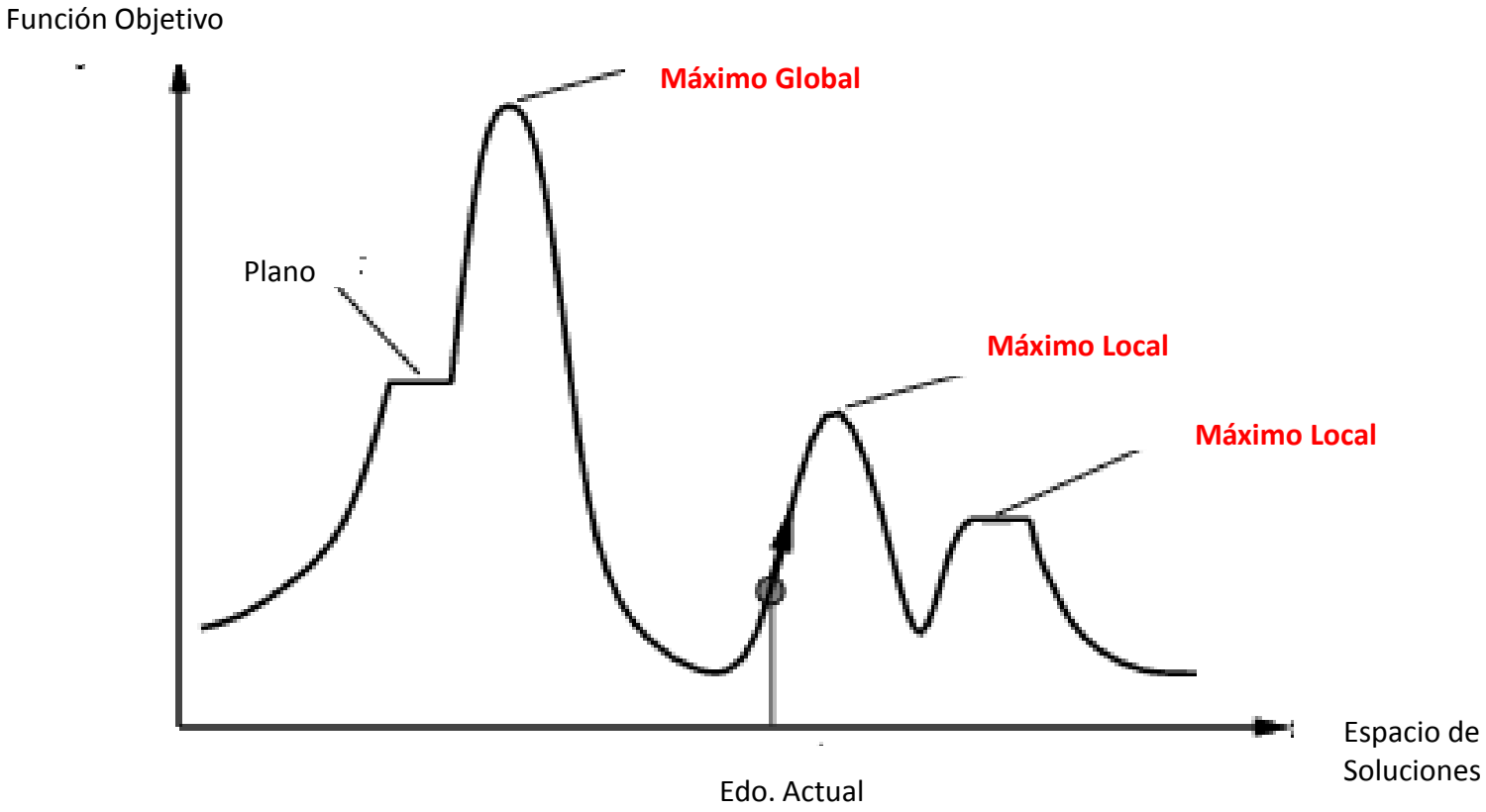
Técnicas de resolución

- Algoritmo Heurística General (**CONTRUCTIVO**)
  - Crear árbol de búsqueda solo con nodo raíz
  - Crear una lista de vecinos al nodo actual para expandir
  - Seleccionar cada nodo de la lista y evaluar solución parcial (uso Función de Evaluación)
  - Escoger para expandir mejor sucesor y repetir si no es edo. objetivo

- Hillclimb (**ITERATIVO**: mejora solución inicial)
  - Actual = Solución-Inicial(problema)
  - Lazo hasta converger
    - próximo= solución vecina a la actual (problema)
    - Si valor(actual) >valor(próximo)
      - actual=próximo

Técnicas de resolución
Búsqueda Local

- Problema: cuidado con los máximos locales



Técnicas de resolución
Primero el Mejor (constructivo)

- Idea: usar una función de evaluación *f(n) p*or nodo
  - Estima que tan "deseable“ es

- Algoritmo:
- Tomar nodo actual
  - Evaluar los nodos sucesores
  - Si alguno es el estado final
    - Detenerse
  - de lo contrario
    - Expandir nodo no expandido mas deseable
  - Regresar al paso inicial hasta no tener a quien expandir

Técnicas de resolución
Primero el Mejor

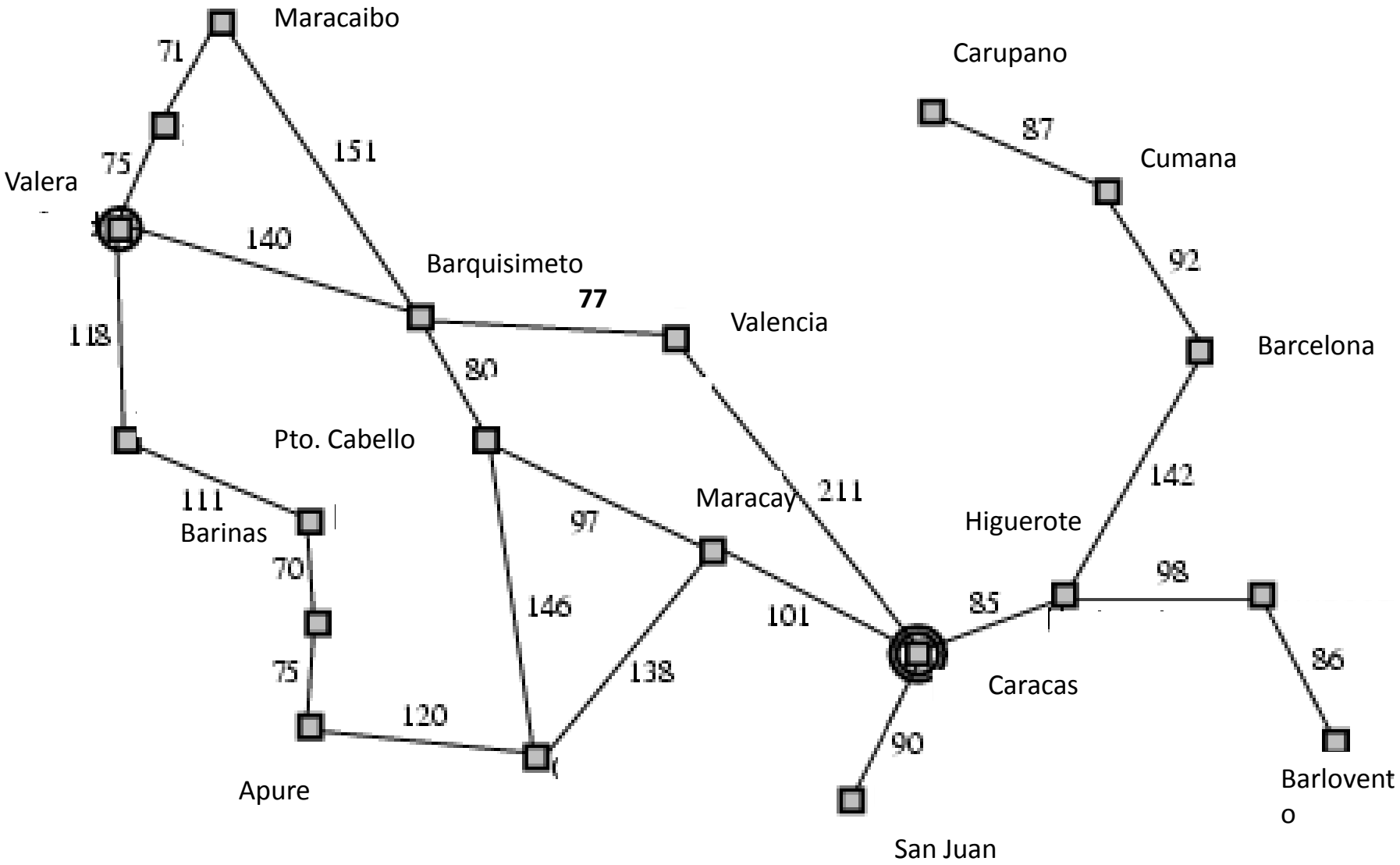
Ejemplo: problema del viajero

- Función de evaluación *f(n)*= estima costo desde *n* hasta *objetivo*

  Por ejemplo: *f(n)* = distancia desde *n* a Caracas

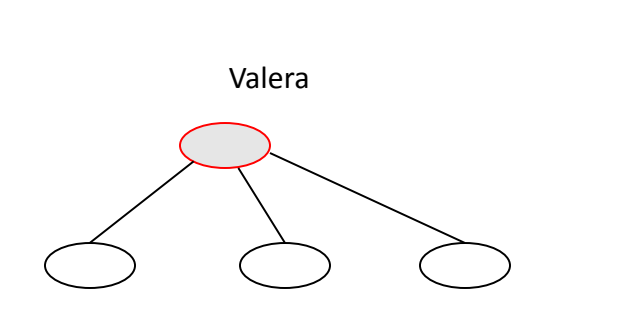
- Algoritmo expande nodo que parece estar mas cerca del objetivo

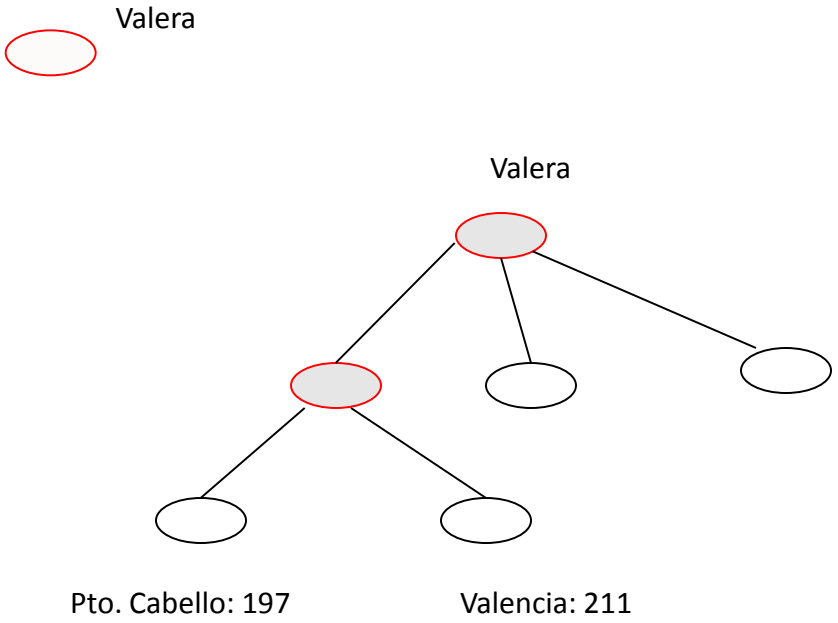
Ejemplo: Venezuela

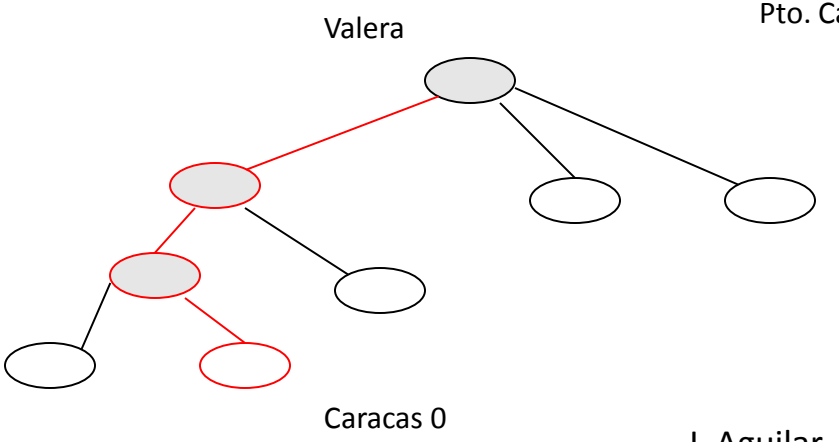


Técnicas de resolución

Solución







Técnicas de resolución

Temple Simulado (iterativo)

- actual =solución_inicial[problema]
- T= alta temperatura
- repita hasta T= congelado
  - próximo= aleatoria selección desde actual de una solución vecina
  - E= costo[próximo]- costo[actual]
  - Si T= congelado y E> 0 entonces
    - regresar actual

parada

Actualiza solución

de lo contrario
- Si E>0 entonces
  - actual= próximo con probabilidad $e^{E/T}$
- Si E< 0 entonces
  - actual= próximo
- descender T si ya han sido aceptados un número dado de movimientos para ese T

Problemas con Restricciones: Constraint satisfaction problems (CSP)

- Componentes:
  - Las variables del problema
  - Los conjuntos posibles de valores de esas variables
  - El conjunto de restricciones que deben ser satisfechas

- Estados definidos por variables X_i con valores desde un conjunto fijo D_i
- Problema de Búsqueda General:
  - Test objetivo definido por un conjunto de restricciones que indican las combinaciones permitidas en los valores de las variables

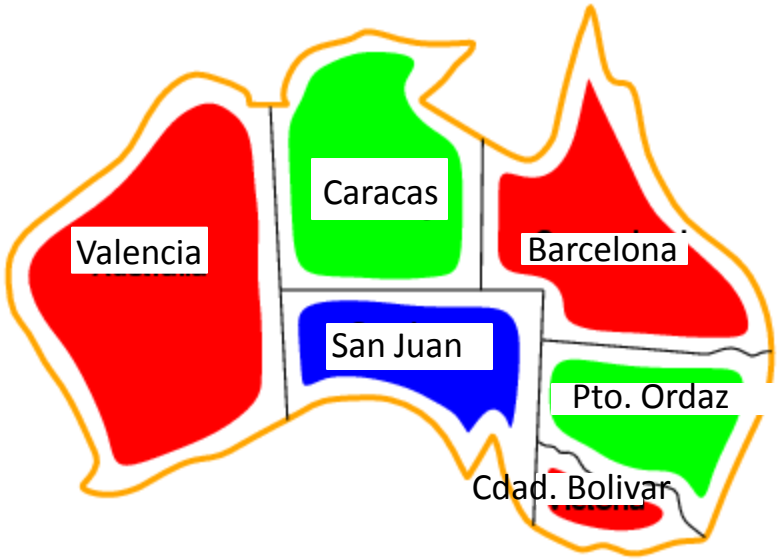
Ejemplo: Coloreado de Mapas

- Variables Valencia, Caracas, San Juan, …
- Dominio D_i = {rojo, verde, azul}
- Restricciones: regiones adyacentes deben tener diferentes colores

- Ejm., Valencia ≠ Caracas,

  => (Valencia, Caracas) puede ser {(rojo,verde),(rojo,azul),(verde,rojo), (verde,azul),(azul,rojo),(azul,verde)}

Ejemplo: Coloreado de Mapas



- Soluciones deben ser completas y consistentes,
- Ejm., Valencia = rojo, Caracas = verde, Barcelona = rojo, Pto. Ordaz = verde, Cda. Bolivar = rojo, San Juan = azul,

PROBLEMAS MULTIOBJETIVOS

- Muchos problemas de optimización se reducen a la consideración de un solo criterio, mientras que en otros problemas es necesario evaluar varios criterios.

problemas multicriterios o multiobjetivos

- Las técnicas de optimización convencional, tales como los métodos basados en gradiente, y unos menos convencionales, tales como recocido simulado, son difíciles de extender a casos verdaderamente multiobjetivos, ya que estos no fueron diseñados para evaluar esto.

PROBLEMAS MULTIOBJETIVOS

En el mundo real, la mayor parte de los problemas tienen varios objetivos (posiblemente en conflicto entre sí) que se desea sean satisfechos de manera simultánea. Muchos de estos problemas suelen convertirse a mono-objetivo transformando todos los objetivos originales, menos uno, en restricciones adicionales.

Hay tres tipos de situaciones que pueden presentarse en un problema multiobjetivo:

- Minimizar todas las funciones objetivo.
- Maximizar todas las funciones objetivo.
- Minimizar algunas funciones y maximizar otras.

PROBLEMAS MULTIOBJETIVOS

- Ejemplo: fabrica de cauchos que deben minimizar dos costos:
  - numero de accidentes y
  - costo de producir cauchos

- Escenarios:

  a = (2,10)

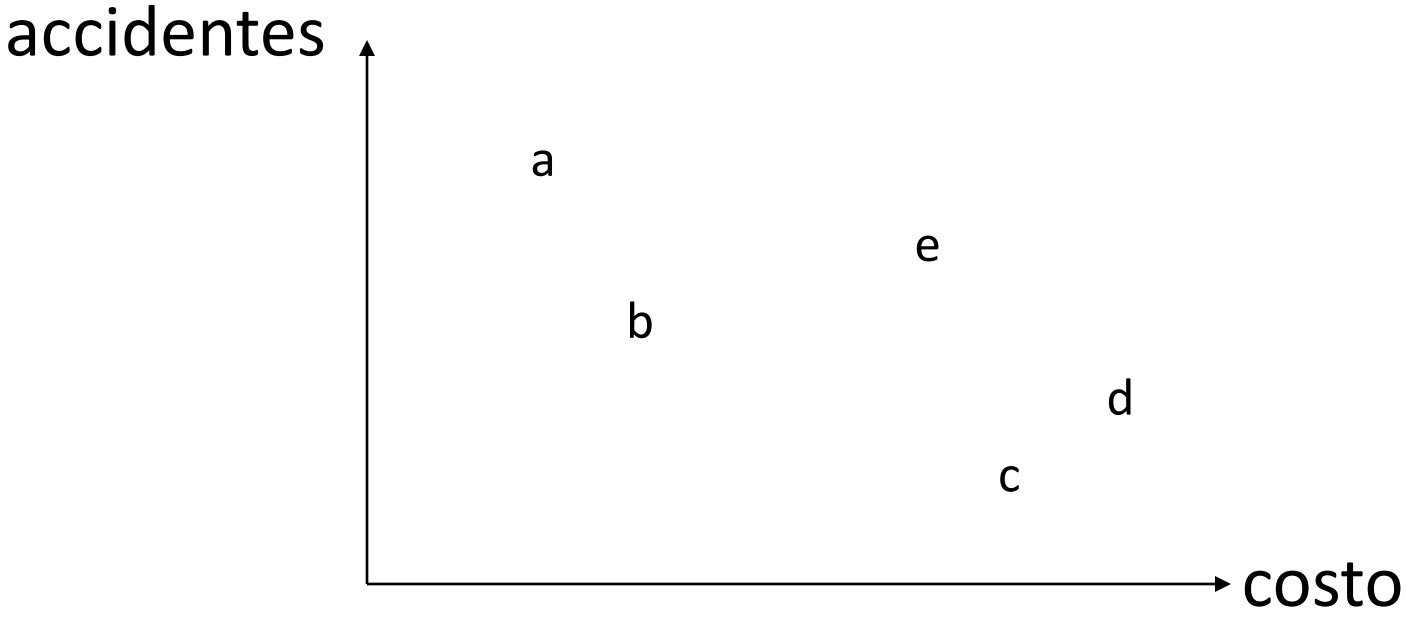
  b = (4,6)

  c = (8,4)

  d = (9,5)

  e = (7,8)

PROBLEMAS MULTIOBJETIVOS



Individuos no dominados: a, b, c!!!

Problemas de Localización Multiobjetivo

DEFINICIÓN:

Escoger un conjunto de lugares para situar varias instalaciones que prestarán su servicio a un conjunto de clientes con el objetivo de optimizar uno o más criterios, económicos o de otra índole.

- La función objetivo es generalmente el costo de operación (fijo y variable)
- En otros contextos es necesario considerar otras funciones objetivo: distancia, cobertura, tiempo de respuesta, equidad, etc.
- Criterios en conflicto
    - Costo
    - Cobertura

Problemas de Localización Multiobjetivo

Información para el modelo

I,i : Conjunto lugares de localización de las instalaciones
m instalaciones

J,j : Conjunto e índice de los lugares de demanda, n clientes

f_i : Costo fijo de operar una instalación en el lugar i.

c_{ij} : Costo de atender toda la demanda del lugar j desde la instalación i.

d_j : Demanda del cliente j.

h_{ij} : Distancia entre i y j

COMPONENTES DEL PROBLEMA DE COBERTURA

D_{max} : Distancia máxima de cobertura.

Q_j : $\{i: h_{ij} \leq D_{max}\}$, conjunto de instalaciones que pueden atender la demanda del nodo j cumpliendo con la distancia máxima de cobertura.

Problemas de Localización Multiobjetivo

Variables de decisión

$$y_i = \begin{cases} 1, & \text{si se abre la bodega } i, \\ 0, & \text{de lo contrario.} \end{cases}$$

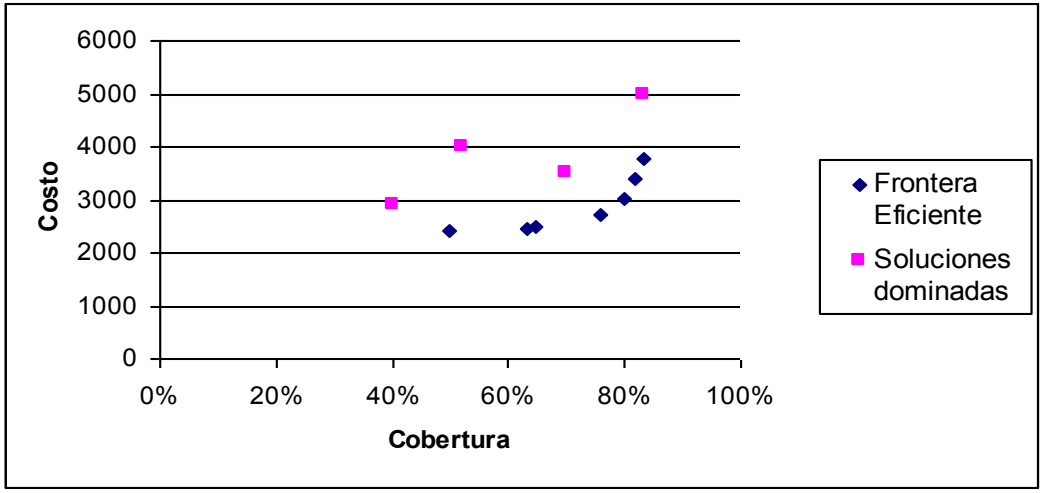
$$x_{ij} = \begin{cases} 1, & \text{si el cliente } j \text{ es atendido por la bodega } i, \\ 0, & \text{de lo contrario.} \end{cases}$$

Problemas de Localización Multiobjetivo

Función objetivo y restricciones

Generalidades Optimización Multiobjetivo

- Los objetivos considerados están en conflicto.
- Se busca un conjunto de soluciones eficientes, en las cuales no sea posible mejorar el valor de una de las funciones objetivo sin deteriorar el desempeño de la otra.



PROBLEMAS MULTIOBJETIVOS

TEORIA DE PARETO

- A diferencia de la optimización de un único objetivo, la solución a un problema multiobjetivo no es un único punto, sino una familia de puntos conocida como el "Conjunto Optimo de Pareto".
- Cada punto en esa superficie es óptimo en el sentido de que no pueden realizarse mejoras en un componente del vector de costo que no conduzca a la degradación en al menos uno de los componentes restantes.
- La noción de optimalidad de Pareto es sólo un primer paso hacia la solución práctica de un problema multiobjetivos; también envuelve la escogencia de una única solución compromiso del "Conjunto Optimo de Pareto" de acuerdo a alguna información de preferencia.

PROBLEMAS MULTIOBJETIVOS

TEORIA DE PARETO

Decimos que un vector de variables de decisión $x^* \in \alpha$ (α es la zona factible) es un óptimo de Pareto si no existe otro $x \in \alpha$ tal que $f_i(x) \leq f_i(x^*)$ para toda $i = 1, \ldots, k$

x^* es un óptimo de Pareto si no existe ningún vector factible de variables de decisión $x^* \in \alpha$ que decremente algún criterio sin causar un incremento simultáneo en al menos un criterio. Desafortunadamente, este concepto casi siempre produce no una solución única sino un conjunto de ellas a las que se les llama conjunto de de Pareto.

Los vectores x^* correspondientes a las soluciones incluidas en el conjunto de óptimos de Pareto son llamados no dominados. La gráfica de las funciones objetivo cuyos vectores no dominados se encuentran en el conjunto de óptimos de Pareto se denominan frente de Pareto.

PROBLEMAS MULTIOBJETIVOS

TEORIA DE PARETO

"Conjunto Optimo de Pareto" esta compuesto por elementos que son soluciones no-inferiores (no-dominados) para el problema multiobjetivos:

Concepto de Inferioridad: Un vector **u** = $(u_1,...,u_n)$ se dice es inferior a **v** = $(v_1,...,v_n)$ si **v** es parcialmente menor que **u** (**v** p < **u**); es decir:

$i = 1,...,n$, $v_i \ u_i$ $\quad i = 1,...,n : v_i < u_i$ (1)

Concepto de Superioridad: Un vector **u** = $(u_1,...,u_n)$ se dice es superior a **v** = (v_1,....,v_n) si **v** es inferior a **u**.