

# Algoritmos de Ramificación y Poda

- Planteamiento General
- Ejemplos de Aplicación
  - Problema de la Mochila entera
  - Problema del Rompecabezas
  - Problema de la Asignación
  - Problema del Viajante

Metodología y Tecnología de la Programación

## Planteamiento General

- Se aplica a problemas que cumplen:
  - Se puedan solucionar mediante algoritmos de vuelta atrás.
  - Se busca una única solución.
- Ejemplos de aplicación;
  - Problema de la mochila 0/1.
  - Problema del rompecabezas.

Metodología y Tecnología de la Programación

## Planteamiento General

- Las funciones de limitación ayudan a eliminar ramas que no llevan a una solución.
- Se definen las funciones de acotación:
  - Eliminan estados basadas en el coste de construcción de una solución (proximidad a una solución).
- La ramificación y poda hace uso de f. de acotación para restringir aún más el conjunto de estados explorados.

Metodología y Tecnología de la Programación

## Planteamiento General

- Las f. de acotación permiten seleccionar el orden en que se visitan los estados.  
*Ramificación y poda = Vuelta atrás + Func. de acotación*
- Diferencia fundamental entre funciones de limitación y func. de acotación:
  - Las primeras deciden si desde un estado se puede alcanzar la solución.
  - Las segundas proporcionan el coste para alcanzar la solución desde un estado.

Metodología y Tecnología de la Programación

## Planteamiento General

### Mochila 0/1

- Sea el problema de la mochila entera:
  - $W = (50, 30, 20, 10)$ ,  $P = (10, 20, 10, 20)$ ,  $M = 60$
  - $S = \{(x_1, x_2, x_3, x_4) : x_i \in \{1,0\}\}$
- Función de limitación:
  - $B_k(X(1), \dots, X(k)) \Leftrightarrow \sum_{i=1}^k X(i) \leq M$
- Función de cota:
  - $Cota(E) = \sum_{i=1}^k X(i) \cdot P(i) + \sum_{i=k+1}^n P(i)$

Metodología y Tecnología de la Programación

## Planteamiento General

### Mochila 0/1

- $Cota(E)$  estima el valor máximo que se puede lograr de  $F$  desde un estado concreto  $E$ .
- Antes de generar un nodo:
  - Se examina el valor de la función de limitación.
  - Se examina el valor de la función de acotación.
  - $c(E) \Leftrightarrow cota(E) > F(Y)$ , siendo  $Y$  el mejor candidato a solución generado hasta el momento.

Metodología y Tecnología de la Programación

## Planteamiento General

### Búsqueda LC

- Diferencia entre generar y explorar.
  - La *visita de un estado* se puede descomponer en dos fases: generar el estado y explorarlo.
  - Un estado se genera si se puede alcanzar solución desde el mismo.
  - La *generación de un estado* consiste en almacenarlo en alguna estructura.
  - La *exploración de un estado* generado consiste en la generación de todos sus posibles hijos.

Metodología y Tecnología de la Programación

## Planteamiento General

### Búsqueda LC

- Estrategias de búsqueda en el árbol de estados.
- La búsqueda en el árbol de estados se realiza por medio de dos estrategias:
  - Exploración en anchura → Mantener en una *cola* los nodos generados pero no explorados.
  - Exploración en profundidad → uso de una *pila*.
- Ninguna de las dos es capaz de explotar la función de acotación.

Metodología y Tecnología de la Programación

## Planteamiento General

### Búsqueda LC

- Uso de una cola de prioridad que tome la función de acotación como valor de prioridad.
  - Exploración de mínimo coste, o LC (Least Cost).
  - Se exploran primero los nodos más prometedores
  - Si se encuentra una solución S tal que  $c(S) > c(E)$ , siendo E el primer candidato de la cola, se finaliza la búsqueda.

Metodología y Tecnología de la Programación

## Planteamiento General

### Acotación basada en coste

- Las func. de acotación se basan en el coste de alcanzar una solución desde un estado.
- Se puede definir el coste  $c(X)$  de alcanzar una solución desde X de dos modos:
  - Número de nodos a generar desde X que es para alcanzar un estado respuesta  $\rightarrow c_1(X)$ .
  - Número de niveles hasta el estado respuesta más cercano desde el nivel de X  $\rightarrow c_2(X)$ .
- Las dos funciones son óptimas.

Metodología y Tecnología de la Programación

## Planteamiento General

### Acotación basada en coste

- Para no explorar el subárbol cuya raíz es  $X$ , se define  $c(X) = f(h(X)) + g(X)$ .
  - $f(\cdot)$  es una función no decreciente.
  - $h(X)$  es el coste de alcanzar  $X$  desde la raíz.
  - $g(X)$  es una estimación del esfuerzo para alcanzar un estado respuesta desde  $X$ . Debe ser no creciente:
    - si  $Y$  es hijo de  $X$ , entonces  $g(X) \geq g(Y)$ .

Metodología y Tecnología de la Programación

## Problema: Rompecabezas de 15 piezas

- Problema del *rompecabezas de 15 piezas*:
  - 15 cuadrados numerados en un marco con capacidad para 16
  - Hay un hueco.
  - La posición objetivo es la de la figura.
  - *Movimientos permitidos* son los que desplazan una pieza adyacente al hueco a dicho hueco, en horizontal o vertical.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Metodología y Tecnología de la Programación

## Problema: Rompecabezas de 15 piezas

- Dada cualquier disposición inicial de las piezas, se pide alcanzar la posición objetivo utilizando sólo movimientos permitidos.
- Cada disposición de las piezas en marco constituye un *estado del rompecabezas*.
- Un estado se alcanza desde el estado inicial si existe una secuencia de movimientos permitidos hacia él.
- El cardinal del espacio de estados es  $16! \approx 20 \cdot 10^{12}$

Metodología y Tecnología de la Programación

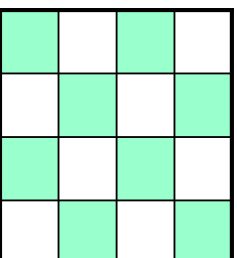
## Problema: Rompecabezas de 15 piezas

- Sólo existe solución desde la mitad de los estados.
  - Sea posición(i) la posición de la pieza i en un estado inicial.
  - Sea menor(i) el número de piezas tales que  $j < i$  y posición(j) > posición(i).

Metodología y Tecnología de la Programación

## Problema: Rompecabezas de 15 piezas

1	3	4	15
2		5	12
7	6	11	14
8	9	10	13



- $menor(1) = 0$
- $menor(4) = 1$
- $menor(12) =$
- Sea  $X = 1$  si en el estado inicial el hueco se halla en una casilla sombreada 0 en caso contrario.
- Entonces el estado objetivo se puede alcanzar desde un estado inicial si  $\sum_{i=1}^{16} menor(i) + X$  es par.

Metodología y Tecnología de la Programación

## Problema: Rompecabezas de 15 piezas

- Los estados alcanzados desde una posición se pueden ver como movimientos del hueco.
- La solución se alcanza realizando los movimientos del hueco: abajo, derecha, abajo.
  - La *exploración en profundidad* nos aleja de la solución más que acercarnos.
  - La *exploración en anchura* genera muchos más estados que los necesarios.

1	2	3	4
5	6		8
9	10	7	11
13	14	15	12

Metodología y Tecnología de la Programación

## Problema: Rompecabezas de 15 piezas

- Podemos definir  $c(X)$  como la longitud del camino desde  $X$  hasta el estado respuesta más cercano  $\rightarrow c(X) = 3$ 
  - Sólo son explorados en una *exploración LC*.
  - No hay forma práctica de calcular  $c(X)$ .
- Se puede definir  $c'(X) = h(X) + g(X)$ .
  - $h(X)$  longitud del camino desde la raíz hasta  $X$ .
  - $g(X) = n^\circ$  de piezas no en su posición objetivo.
  - $g(X)$  es una cota inferior del  $n^\circ$  de movimientos.

Metodología y Tecnología de la Programación

## Planteamiento General Esquema de ramificación y poda

- $c(X)$  se estima por medio de una función  $c'(X)$  que cumple las condiciones siguientes:
  - $c'(X)$  se puede calcular en cada momento utilizando sólo información local al nodo.
  - $c'(X) = c(X)$  para todo estado respuesta  $X$ .
- $c'(X)$  permite encontrar los nodos respuesta usando una estrategia de exploración LC.

Metodología y Tecnología de la Programación

## Planteamiento General

### Esquema de ramificación y poda

- Se necesita disponer de una cola de prioridad.
- Para cada estado, se almacena:
  - El valor de la función  $c'(X)$  para que se mantenga ordenada por esta cantidad.
  - La información necesaria para poder reconstruir la solución.

Metodología y Tecnología de la Programación

## Planteamiento General

### Esquema de ramificación y poda

```
procedimiento LC(raíz: estado, coste: función)  
var cola: cola de prioridad de estado;  
actual ← raíz  
cola ← vacía  
respuesta ← falso  
repetir  
  para cada X ∈ hijos(actual)  
    hacer X.padre ← actual  
    si respuesta(X)  
      entonces respuesta ← cierto  
        escribir(X)  
        sino insertar(X, cola, coste)  
    hasta respuesta o vacía(cola)  
  si no(respuesta)  
    entonces escribir('No respuesta')
```

Metodología y Tecnología de la Programación

## Planteamiento General

### Esquema de ramificación y poda

- Las funciones utilizadas en el algoritmo anterior son las siguientes:
  - hijos(X) devuelve los hijos no limitados de X.
  - respuesta(X), cierto si X es un estado respuesta.
  - escribir(X) reconstruye una solución a partir del estado respuesta X.
  - insertar(X,C,coste), primero(C), vacía(C) son las operaciones de uso de la cola C.

Metodología y Tecnología de la Programación

## Planteamiento General

### Esquema de ramificación y poda

- El algoritmo LC devuelve respuesta de menor coste o explora todo el árbol si ésta no existe.
  - Supongamos que  $c'(X)$  está elegida de modo que se cumple que  $c'(Y) < c'(Z)$  si y sólo si  $c(Y) < c(Z)$ .
  - Entonces el algoritmo LC termina y devuelve el estado de mínimo coste E.
- Difícil encontrar  $c'(X)$  que cumpla lo anterior.
  - Fácil que cumpla que  $c'(X) = c(X)$  para todo estado respuesta y  $c'(X) \leq c(X)$  para el resto.

Metodología y Tecnología de la Programación

## Planteamiento General

### Esquema de ramificación y poda

- El algoritmo LC no siempre encuentra el estado respuesta de menor coste.
- Es preciso modificar el algoritmo LC.
  - En LC, se devuelve el primer estado respuesta generado.
  - Sólo se debe devolver un estado respuesta cuando se explora.