

# Ramificación y Poda

La Ramificación y poda se suele utilizar en problemas de optimización discreta y en problemas de juegos

La ramificación y poda puede ser visto como una generalización o mejora del BACKTRACKING.

## *Ramificación y Poda / Backtracking*

- En Backtracking, en cuanto se genera un nuevo nodo hijo pasa a procesarse
- En Ramificación y Poda, se generan todos los nodos hijos del nodo actual y después se van procesando
- En Backtracking los únicos nodos vivos son los que están en el camino desde la raíz hasta el nodo que se está estudiando
- En Ramificación y Poda puede haber más nodos vivos que se almacenan en la lista de nodos vivos
- El recorrido no tiene que ser necesariamente en profundidad en RyP.

## Esquema algorítmico de ramificación y poda

- **Inicialización:** Meter la raíz en la LNV, e inicializar la variable de poda **C** de forma conveniente.
- **Repetir** mientras no se vacíe la LNV:
  - **Sacar un nodo** de la LNV, según la estrategia de ramificación.
  - Comprobar si debe ser podado, según la **estrategia de poda**.
  - En caso contrario, **generar sus hijos**. Para cada uno:
    - Comprobar si es una **solución final** y tratarla.
    - Comprobar si debe ser **podado**.
    - En caso contrario, **meterlo en la LNV** y **actualizar C** de forma adecuada.

## ramificacionYpoda(Tipo\_Nodo: raiz, Tipo\_Nodo: s)

```
1 LNV = {raiz}
2 C = CotaSuperior(raiz)
3 S = {∅}
4 ( LNV ≠ {∅} ) [
    X = seleccionar(LNV) // estrategia Ramificación
    LNV = LNV - {X}
    si (CotaInferior(X) < C) entonces //PODA
        Para Cada Hijo Y de X Hacer
            SI (solucion (Y) y Valor (Y) < valor (S) ) ento
                S = Y
                C = min(C, Valor (Y))
            sino
                si ( ! Solucion (Y) y CotaInferior(Y) < C ) entonces
                    LNV = LNV + {Y}
                    C = min(C, CotaSuperior(Y))
                FinSi
            FinSi
        FinPara
    ]
```

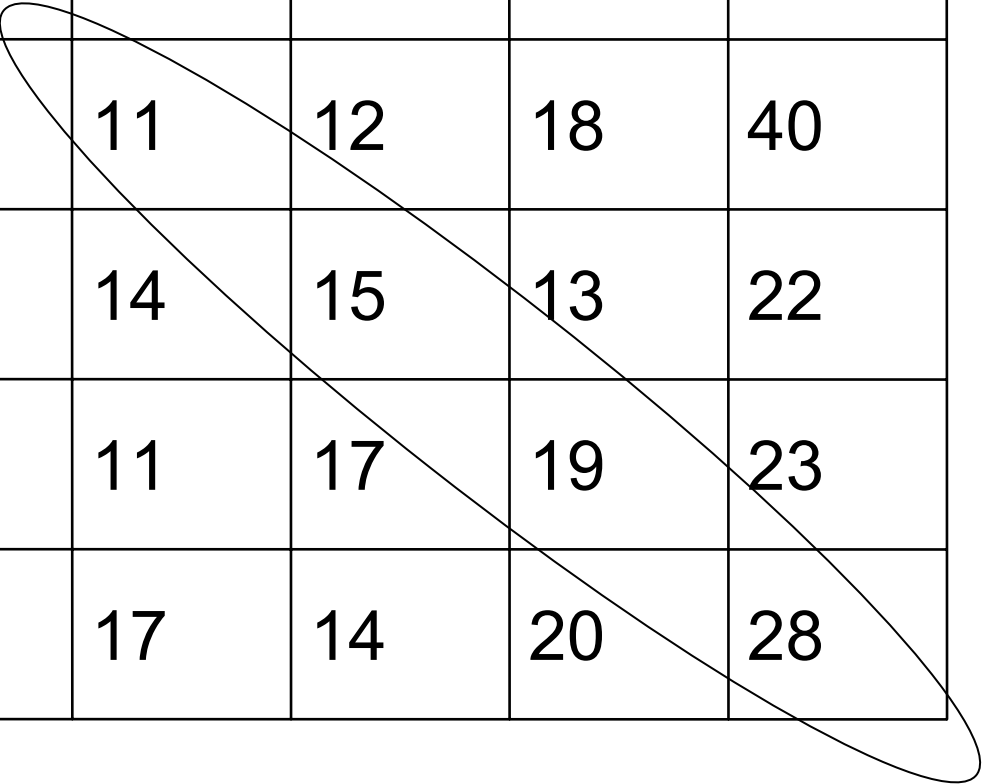
-LNV: lista de nodos vivos  
-CotaSuperior():Cota superior del beneficio óptimo  
-seleccionar(): Método de selección del prox nodo a probar.  
-solucion():Comprobar si es una solución final y tratarla

# Ejemplo

## TAREAS

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

AGENTES



~~LS = 73~~

LS = 64

60 (11 + 14 + 13 + 22)

68 (12 + 14 + 19 + 23)

64 (12+13+11+28)

A->1

~~B->1~~

C->1 D-> 4

58 (12 + 11 + 13 + 22)

59 (12 + 13 + 11 + 23)

A->2

B->3

65 (12+13+23+17)

65 (18 + 11 + 14 + 22)

64 (12 + 22 + 11 + 19)

~~C->4 D->1~~

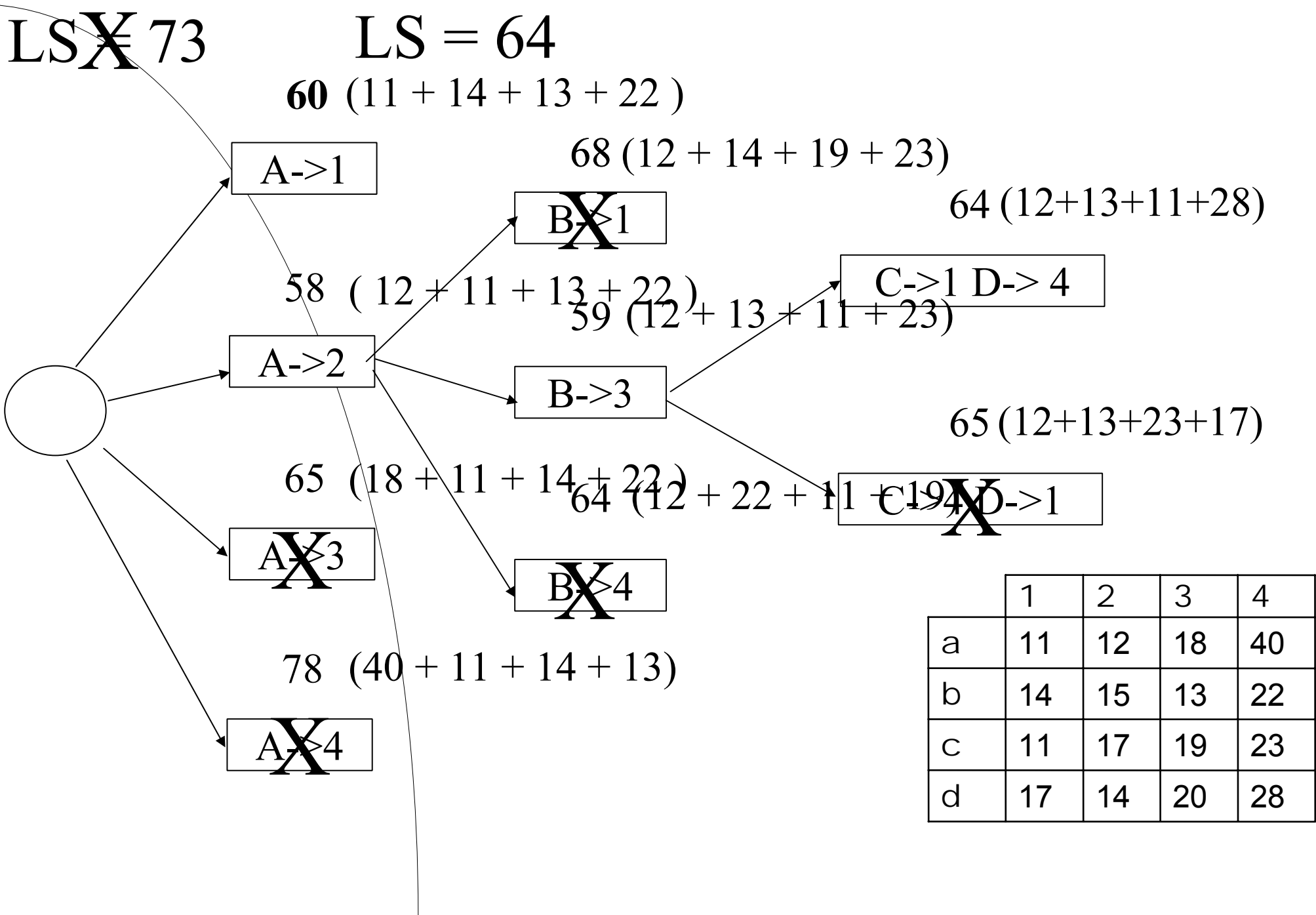
~~A->3~~

~~B->4~~

78 (40 + 11 + 14 + 13)

~~A->4~~

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28



~~LS = 64~~

LS = 61

68 (11+15+19+23)

69 (11+13+17+28)

~~B > 2~~

~~C -> 2 D -> 4~~

61 (11+13+14+23)

61 (11+13+23+14)

A -> 1

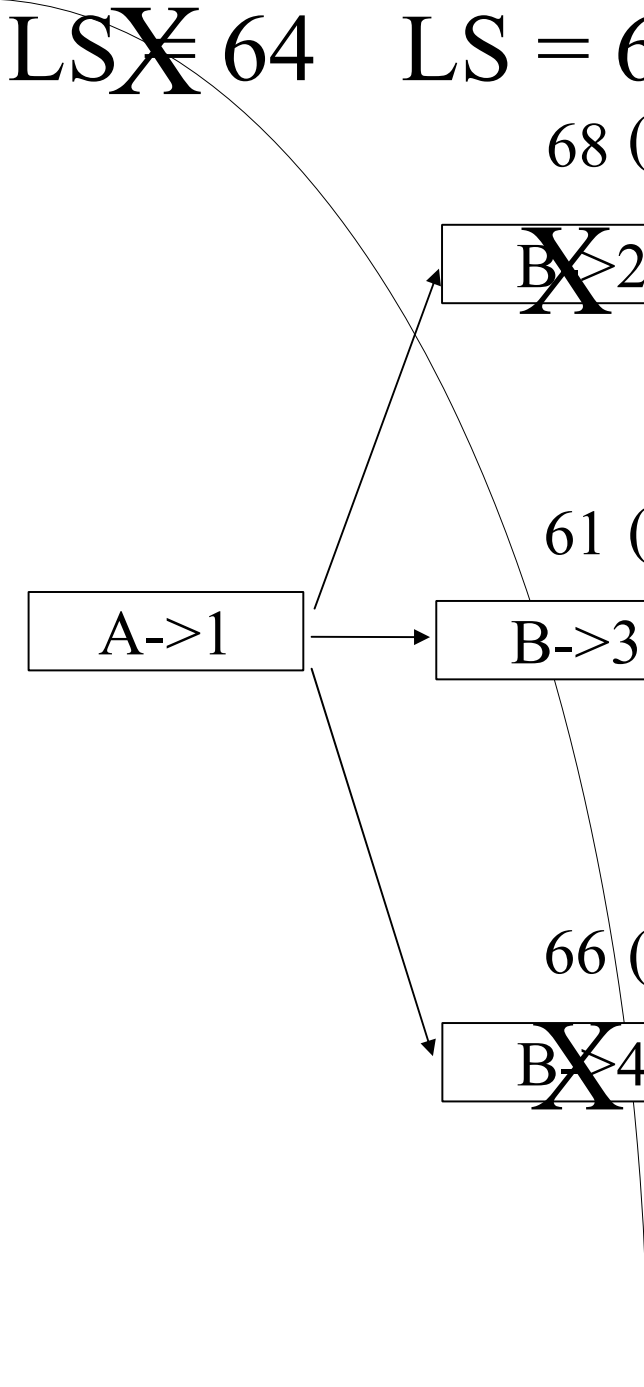
B -> 3

C -> 4 D -> 2

66 (11+22+14+19)

~~B > 4~~

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28





The diagram consists of a grid of numbers. On the left side, there is a vertical line that curves at the top and ends in a small hook. The grid is composed of four rows and four columns. The first row contains the numbers 1, 2, 3, and 4. The second row contains the letters a, 11, 12, 18, and 40. The third row contains the letters b, 14, 15, 13, and 22. The fourth row contains the letters c, 11, 17, 19, and 23. The fifth row contains the letters d, 17, 14, 20, and 28.

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

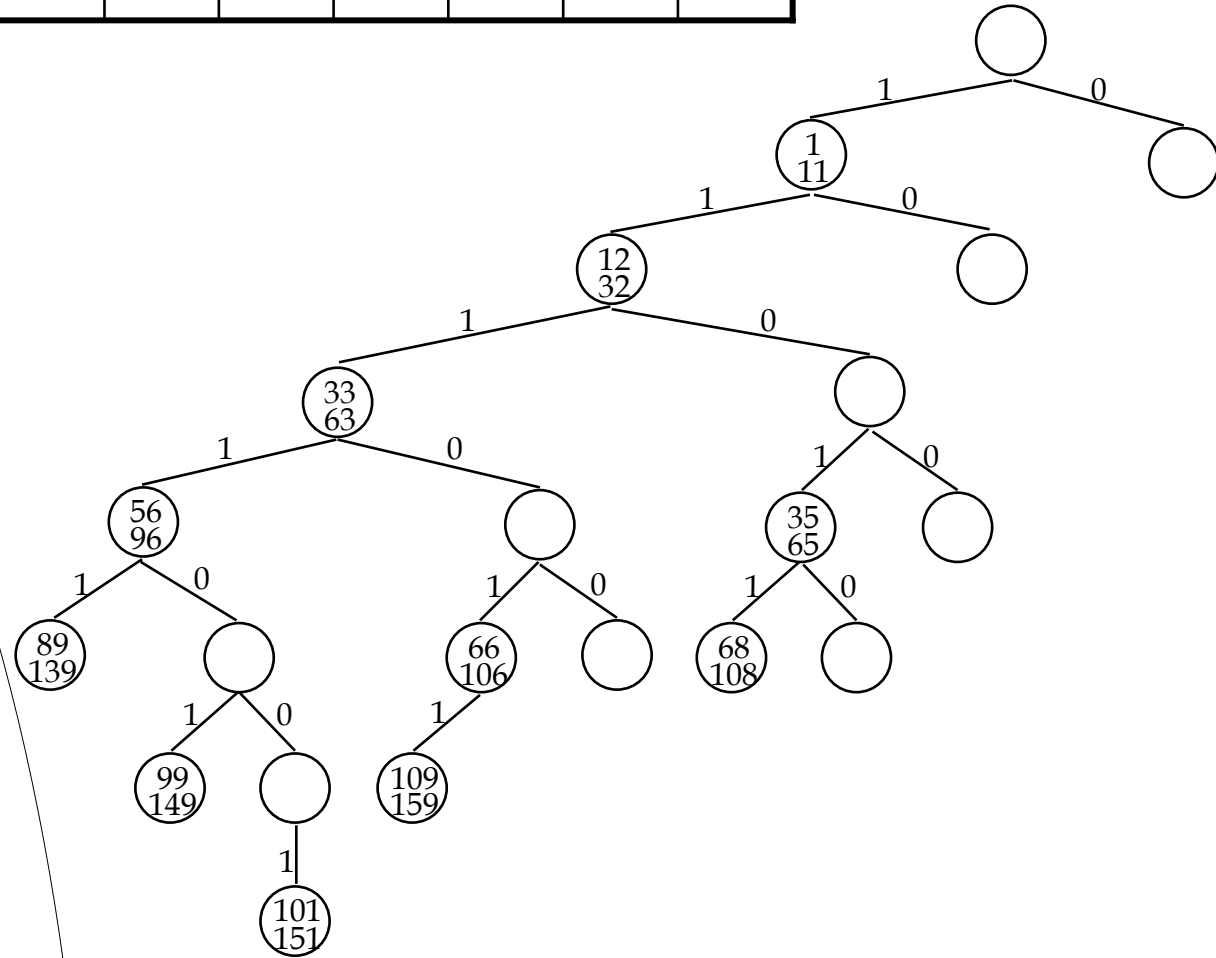
# El Problema de la mochila Por Ramificación y Poda

Maximizar  $\sum x_i v_i$  con la restricción de  $\sum x_i w_i < W$ , donde  $v$  y  $w$  son positivos y los  $X$  son enteros positivos. Supongamos que las variables están ordenadas de la siguiente forma:  $v_i / w_i \geq v_{i+1} / w_{i+1}$

Beneficio	11	21	31	33	43	53	55	65
Peso	1	11	21	23	33	43	45	55

Capacidad = 110

N = 8



ben=159

sol=(1,1,1,0,1,1,0,0)