Técnica de diseño #5:
Branch and bound

Análisis y Diseño de Algoritmos
Ing. Román Martínez M.

Branch and bound

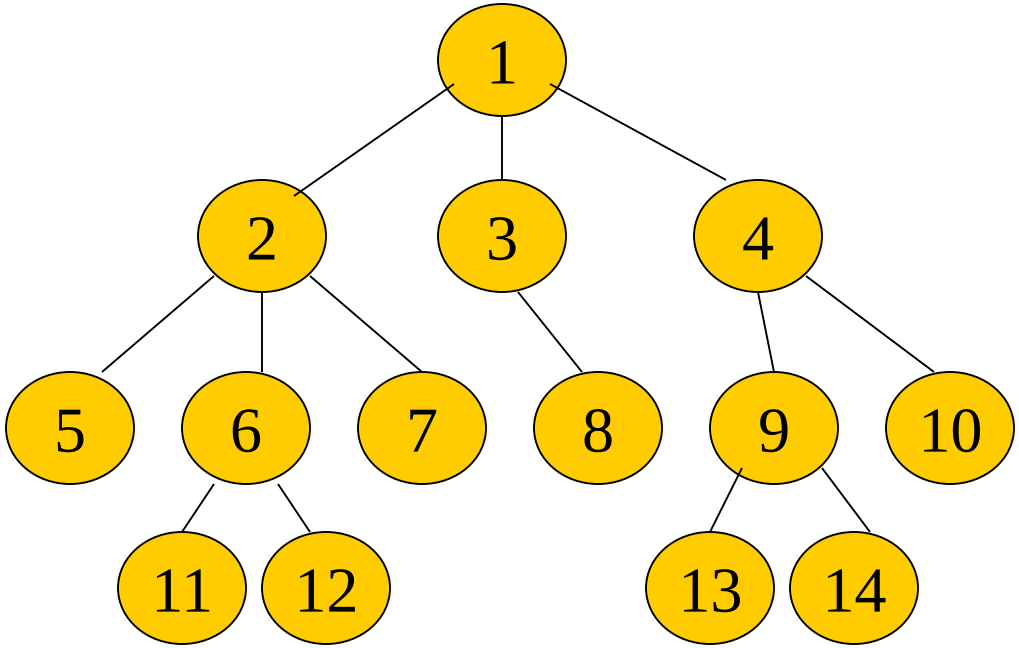
- **Branch and bound** es una técnica muy similar a la de **Backtacking**, pues basa su diseño en el análisis del árbol de búsqueda de soluciones a un problema.
- Sin embargo, no utiliza la búsqueda en profundidad (*depth first*), y sólamente se aplica en problemas de OPTIMIZACIÓN.
- Los algoritmos generados por está técnica son normalmente de orden exponencial o peor en su peor caso, pero su aplicación ante instancias muy grandes, ha demostrado ser eficiente (incluso más que bactracking).

Diseño de algoritmos con Branch and bound

- Decidir de qué manera se conforma el árbol de búsqueda de soluciones.
- Sobre el árbol, se aplicará una búsqueda en anchura (*breadth-first*), pero considerando prioridades en los nodos que se visitan (*best-first*).
- El criterio de selección de nodos, se basa en un **valor óptimo posible** (bound), con el que se toman decisiones para hacer las podas en el árbol.

Recordando... recorrido *breadth first*

- En un árbol, corresponde al recorrido NIVEL POR NIVEL…



- Y, ¿cómo se implementa el algoritmo?

Algoritmo del recorrido NIVEL x NIVEL

```
void nivelXnivel (Nodo r)
{  Nodo n, h; Fila f;
   Meter en f al apuntador r;
   while (f no esté vacía)
   { Sacar de f en n;
      Visitar n;
      for (cada hijo h de n)
         Meter en f al apuntador h; }
}
```

Este será el esqueleto general de los algoritmos con Branch and bound

El problema de la mochila

- El problema ya fue resuelto con Programación dinámica, con Backtracking, y ahora se hará con Branch and bound.
- La solución con Backtracking, ya consideró que era un problema de optimización, e incluyó en su criterio de selección de nodos, el proceso de calcular un valor óptimo posible (valor posible a acumular), para tomar las decisiones de podas…
- Esta estrategia servirá pero ahora en el contexto de Branch and bound, lo cual implica que se realizará la búsqueda nivel por nivel en vez de primero en profundidad.

Problema de la mochila

- Arbol de búsqueda de soluciones:
  - Similar al del problema de "Sum-of-subsets"…
  - Cada nivel indica un objeto a incluir en la mochila…
  - Cada nodo del árbol tiene 2 hijos; uno que incluye al objeto y otro que no lo incluye…
- Criterio de selección:
  - Si el peso acumulado de los objetos incluídos no excede a la capacidad de la mochila…
  - Si el valor posible a acumular es mayor al mejor valor acumulado hasta ese momento...

Valor óptimo posible (valor posible a acumular)

- **Análisis idéntico a lo que se hizo con backtracking:**
  - Sea el nodo del nivel k el que hace que el peso acumulado exceda al peso permitido…
  - El valor posible a acumular, se puede calcular de la siguiente forma:
    - ✓ Valor acumulado por los objetos ya incluidos, más…
    - ✓ Valor de los objetos i+1 hasta k-1, más…
    - ✓ Valor proporcional del objeto k por la fracción de peso que resta en la mochila…
  - La fracción de peso que resta en la mochila se puede calcular:
    - ✓ Peso permitido en la mochila, menos…
    - ✓ Peso acumulado por los objetos ya incluídos, menos…
    - ✓ Peso de los objetos i+1 hasta k-1.

Ejemplo

- Para una mochila con capacidad de soportar 16 unidades de peso, se tienen los siguientes 4 objetos:
  - $Objeto_1$, $valor_1$: \$40, $peso_1$: 2, $valor_1/peso_1$ = \$20
  - $Objeto_2$, $valor_2$: \$30, $peso_2$: 5, $valor_2/peso_2$ = \$6
  - $Objeto_3$, $valor_3$: \$50, $peso_3$: 10, $valor_3/peso_3$ = \$5
  - $Objeto_4$, $valor_4$: \$10, $peso_4$: 5, $valor_4/peso_4$ = \$2

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2

Va = \$0
Pa = 0
Vp = \$115

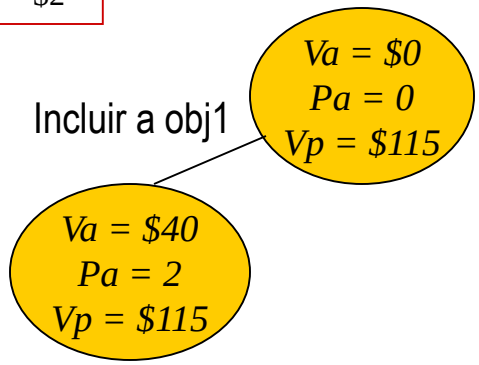
Valor óptimo = \$0

✓ **Pacum < 16**
✓ **Vposible > Valor óptimo**

Valor posible a acumular:
Se pueden acumular los objetos 1 y 2 sin exceder el peso.
\$40+\$30+(16-2-5)*\$50/10 = \$115

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



Valor óptimo = \$40

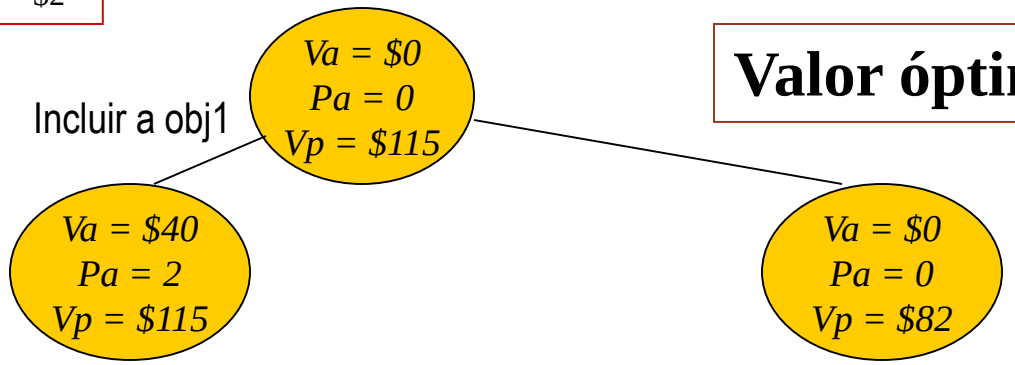
- ✓ **Pacum < 16**
- ✓ **Vposible > Valor óptimo**

Valor posible a acumular:

Se pueden acumular el objeto 2 sin exceder el peso.
\$40+\$30+(16-2-5)*\$50/10 = \$115

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



Valor óptimo = \$40

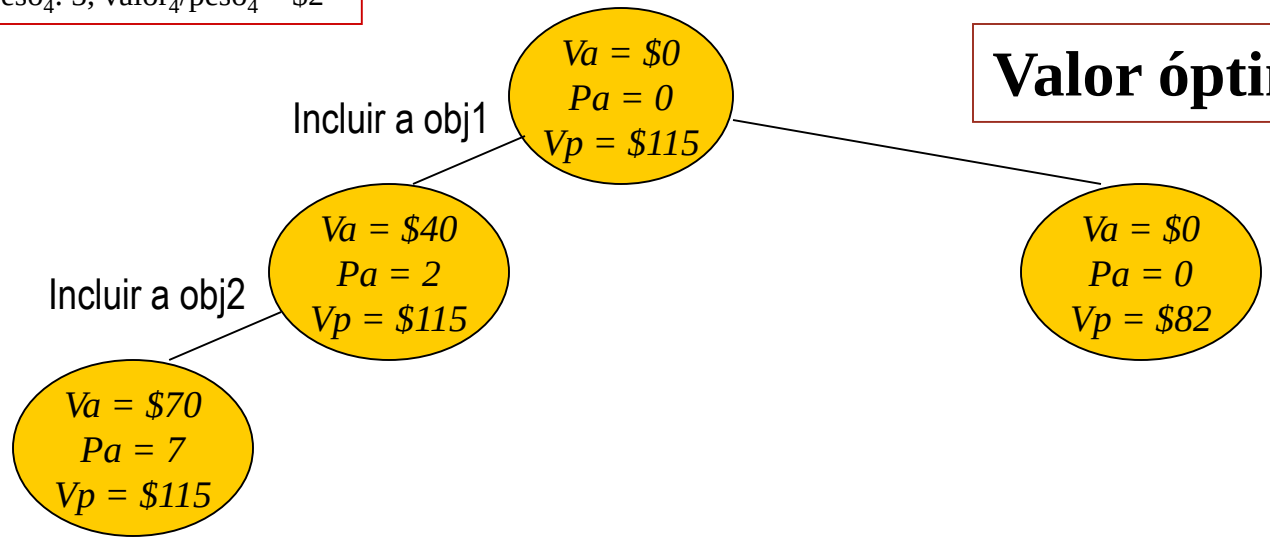
- ✓ **Pacum < 16**
- ✓ **Vposible > Valor óptimo**

Valor posible a acumular:

Se pueden acumular el objeto 2 y 3 sin exceder el peso.
\$30+\$50+(16-5-10)*\$10/5 = \$82

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



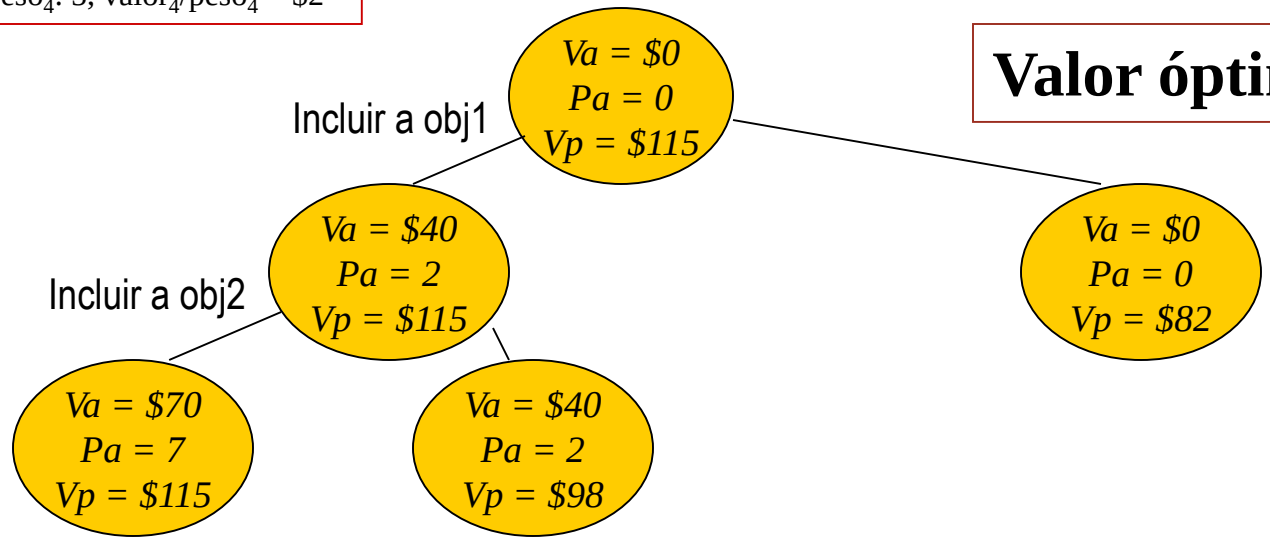
Valor óptimo = \$70

- ✓ **Pacum < 16**
- ✓ **Vposible > Valor óptimo**

Valor posible a acumular:
NO se pueden acumular más objetos sin exceder el peso.
\$70+(16-7)*\$50/10 = \$115

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



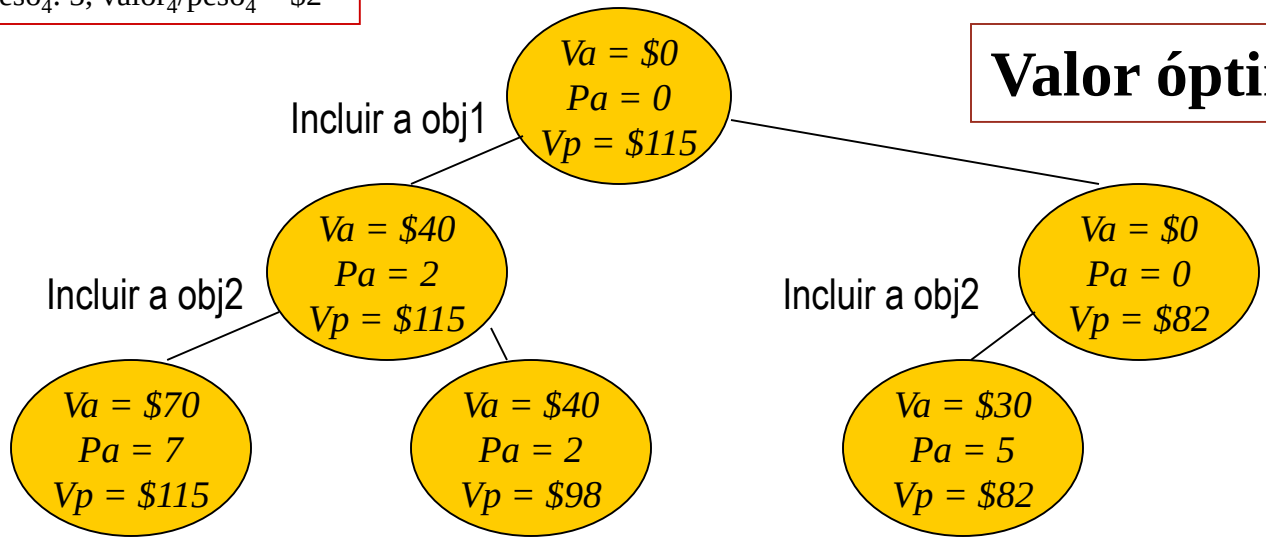
Valor óptimo = \$70

- ✓ **Pacum < 16**
- ✓ **Vposible > Valor óptimo**

Valor posible a acumular:
Se puede incluir el objeto 3 sin exceder el peso
\$40+\$50+(16-2-10)*\$10/5 = \$98

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



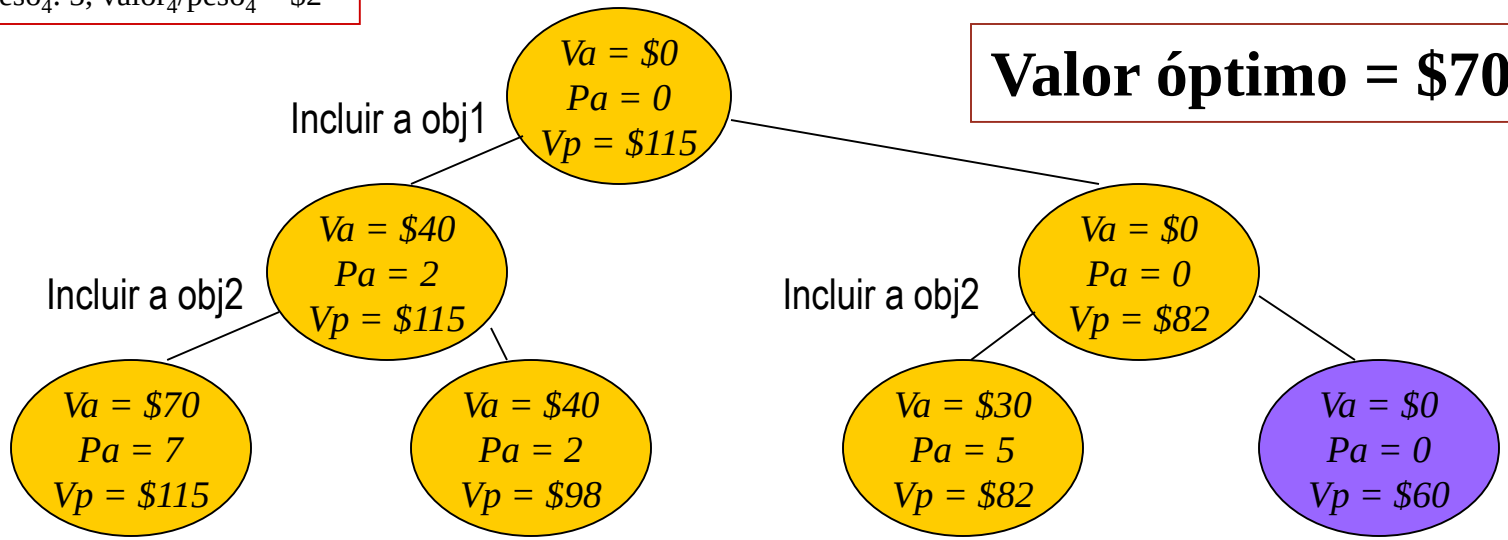
Valor óptimo = \$70

- ✓ **Pacum < 16**
- ✓ **Vposible > Valor óptimo**

Valor posible a acumular:
Se puede incluir el objeto 3 sin exceder el peso
\$30+\$50+(16-5-10)*\$10/5 = \$82

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



Valor óptimo = \$70

✓ **Pacum < 16**

× **Vposible > Valor óptimo**

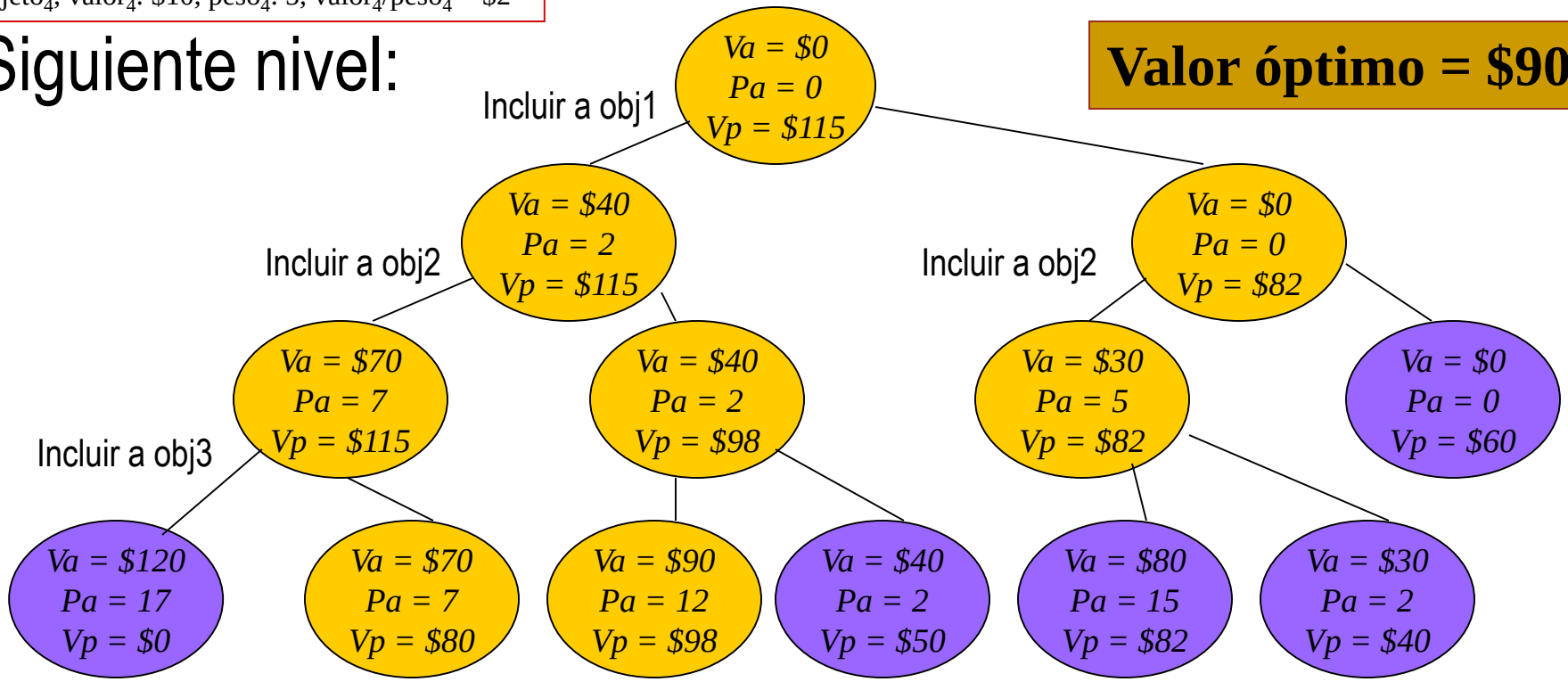
Valor posible a acumular:
Se pueden incluir los objeto 3 y 4 sin exceder el peso
\$50+\$10+(16-5-10)*\$0 = \$60

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2

Siguiente nivel:

Valor óptimo = \$90

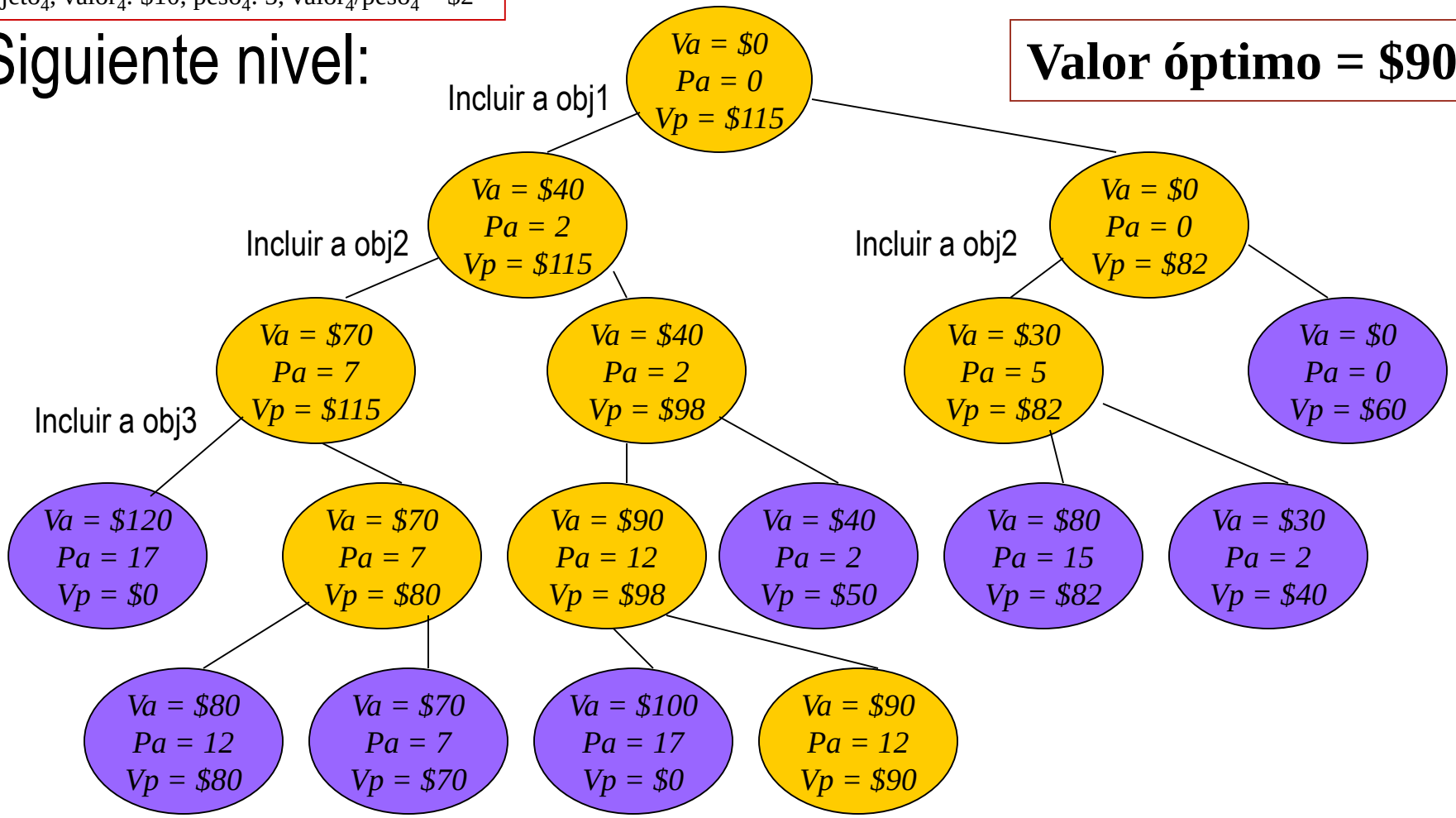


Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2

Siguiente nivel:

Valor óptimo = \$90

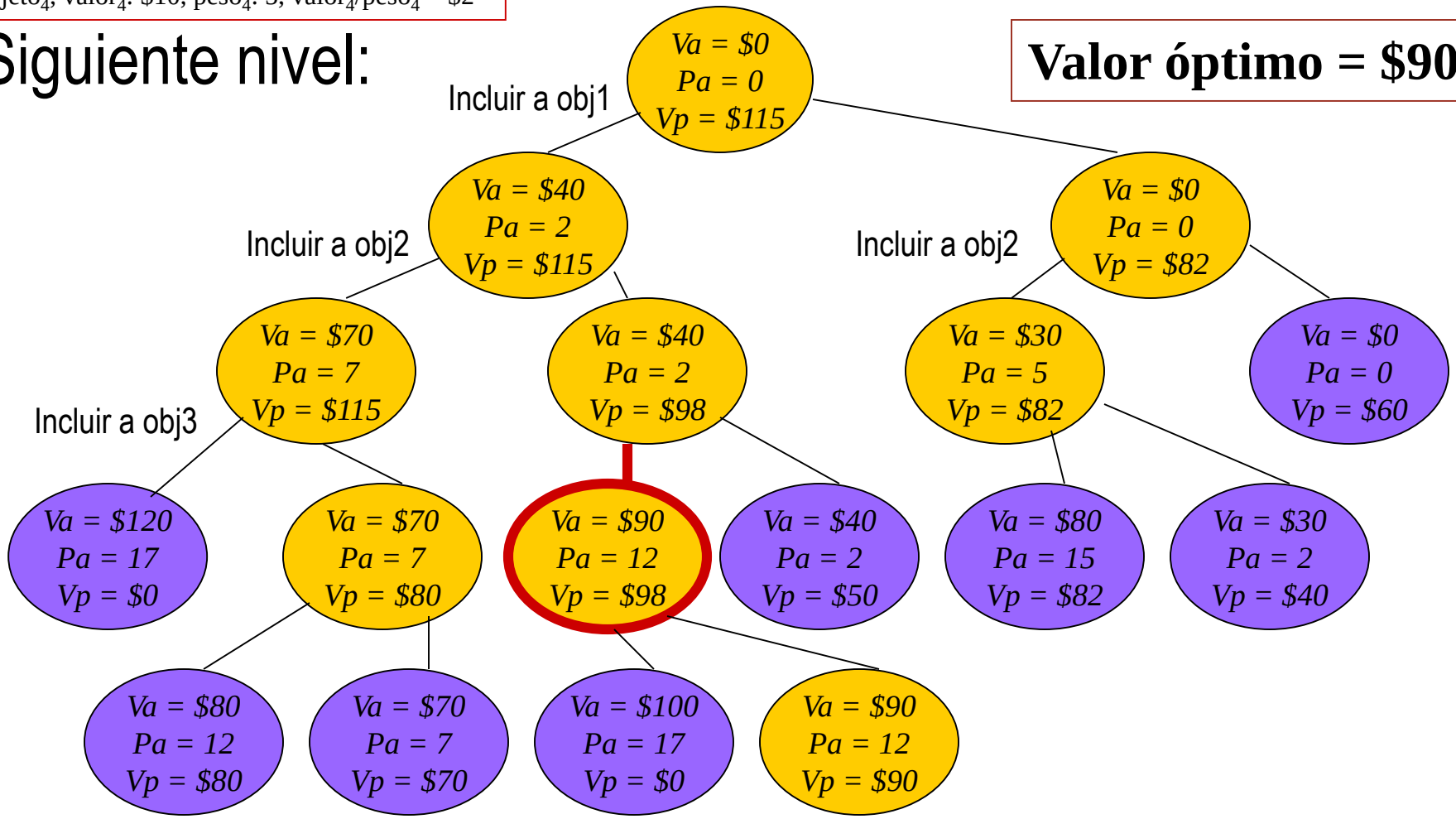


Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2

Siguiente nivel:

Valor óptimo = \$90



Análisis del ejemplo

- El árbol de búsqueda de solución por Backtacking analizó 13 nodos…
- Por Branch and bound analizó 17 nodos…
- Aparentemente NO hay beneficio…
- Por lo tanto, se tiene que mejorar la técnica…
- ¿Cómo?
- Aprovechando la visita de todo un nivel, escoger la expansión del "mejor nodo" (*best-first*)...

Ejemplo

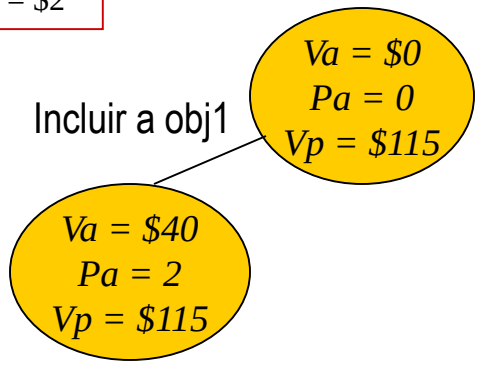
PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2

Va = \$0
Pa = 0
Vp = \$115

Valor óptimo = \$0

Ejemplo

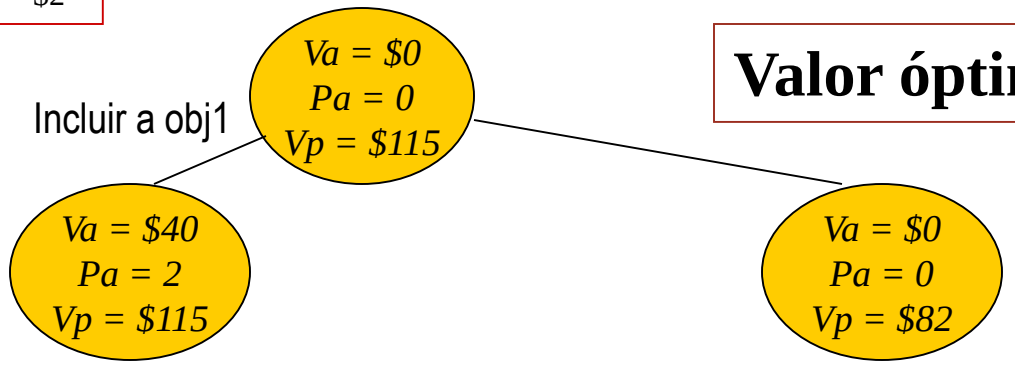
PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



Valor óptimo = \$40

Ejemplo

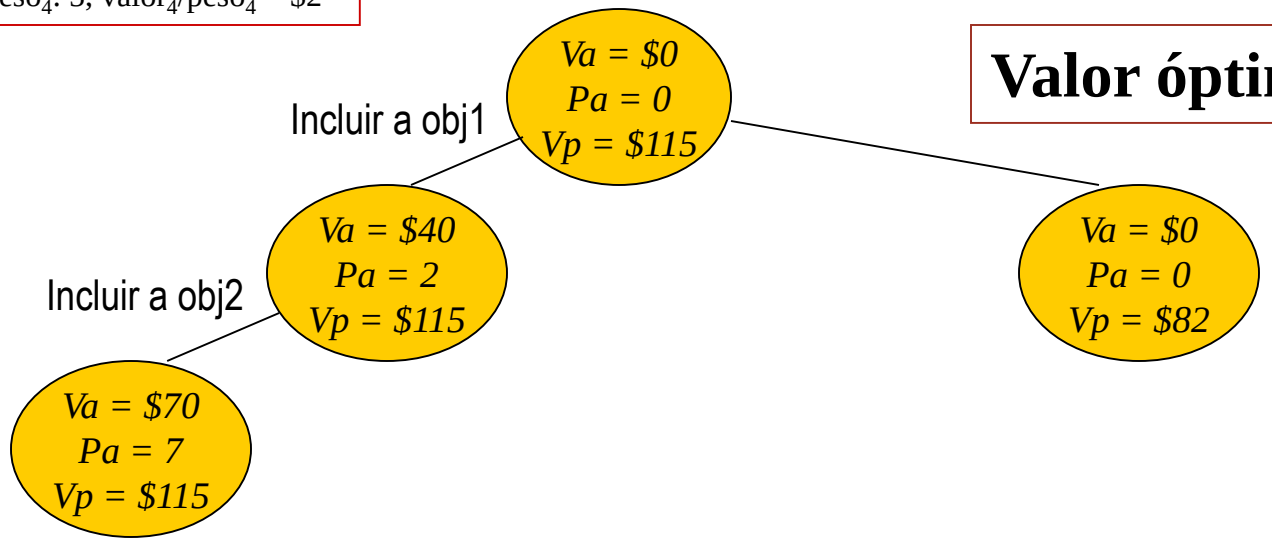
PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



Valor óptimo = \$40

Ejemplo

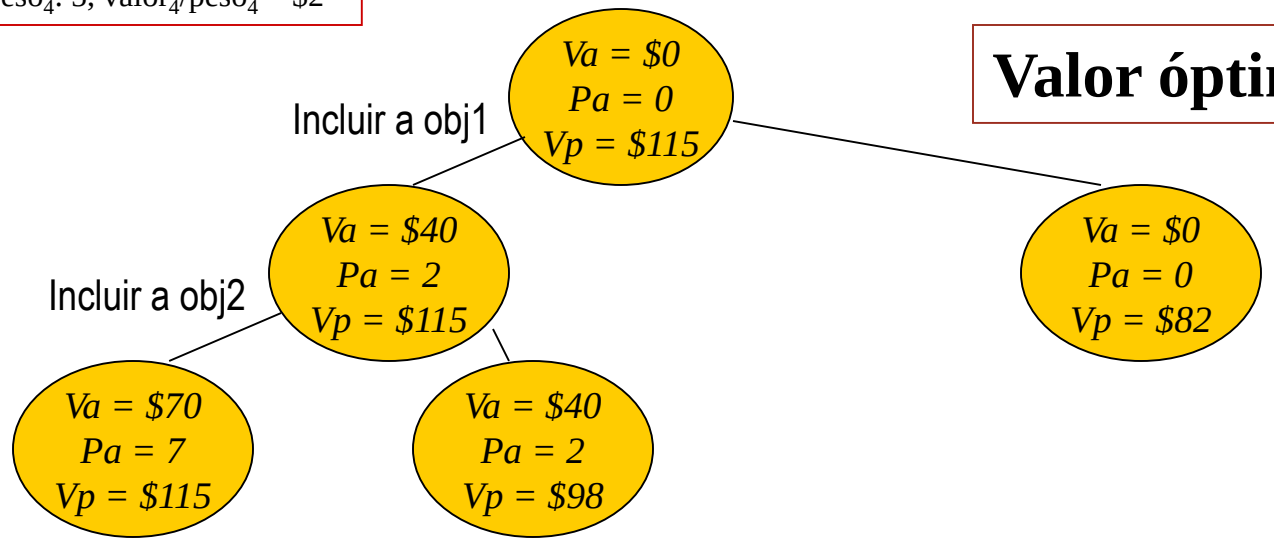
PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



Valor óptimo = \$70

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2

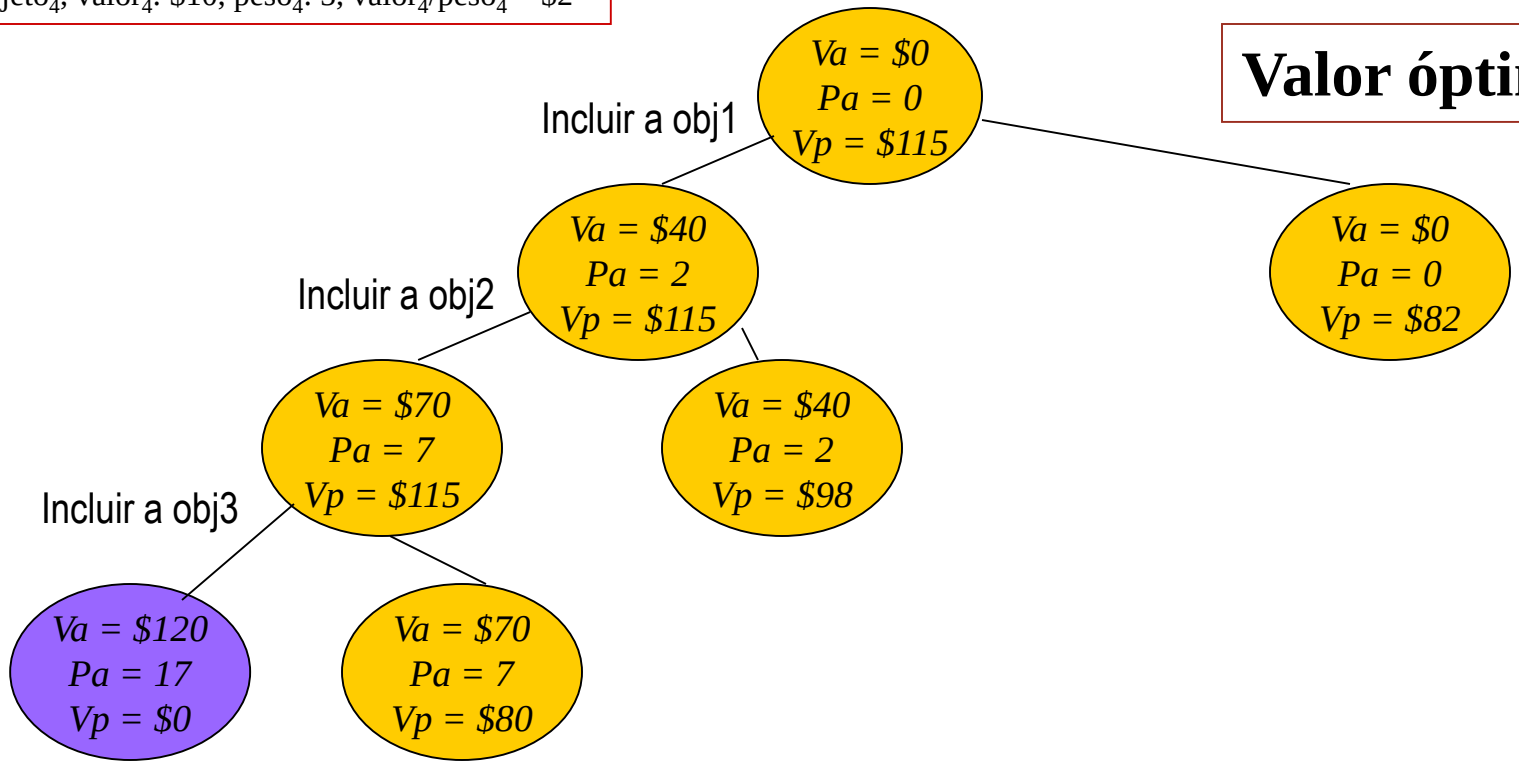


Valor óptimo = \$70

De los nodos a expandir,
¿cuál tiene el mejor valor posible?

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2

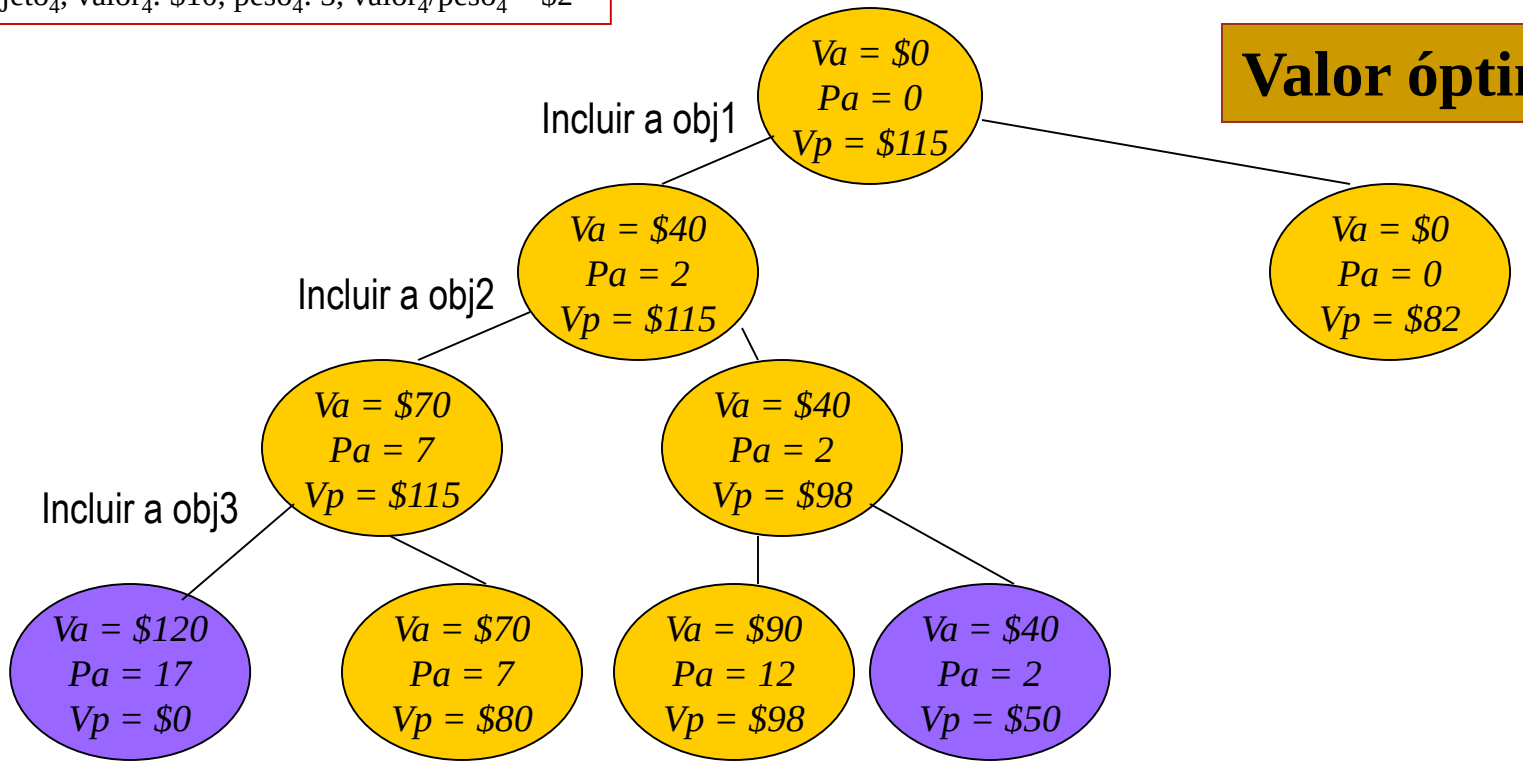


Valor óptimo = \$70

De los nodos a expandir,
¿cuál tiene el mejor valor posible?

Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



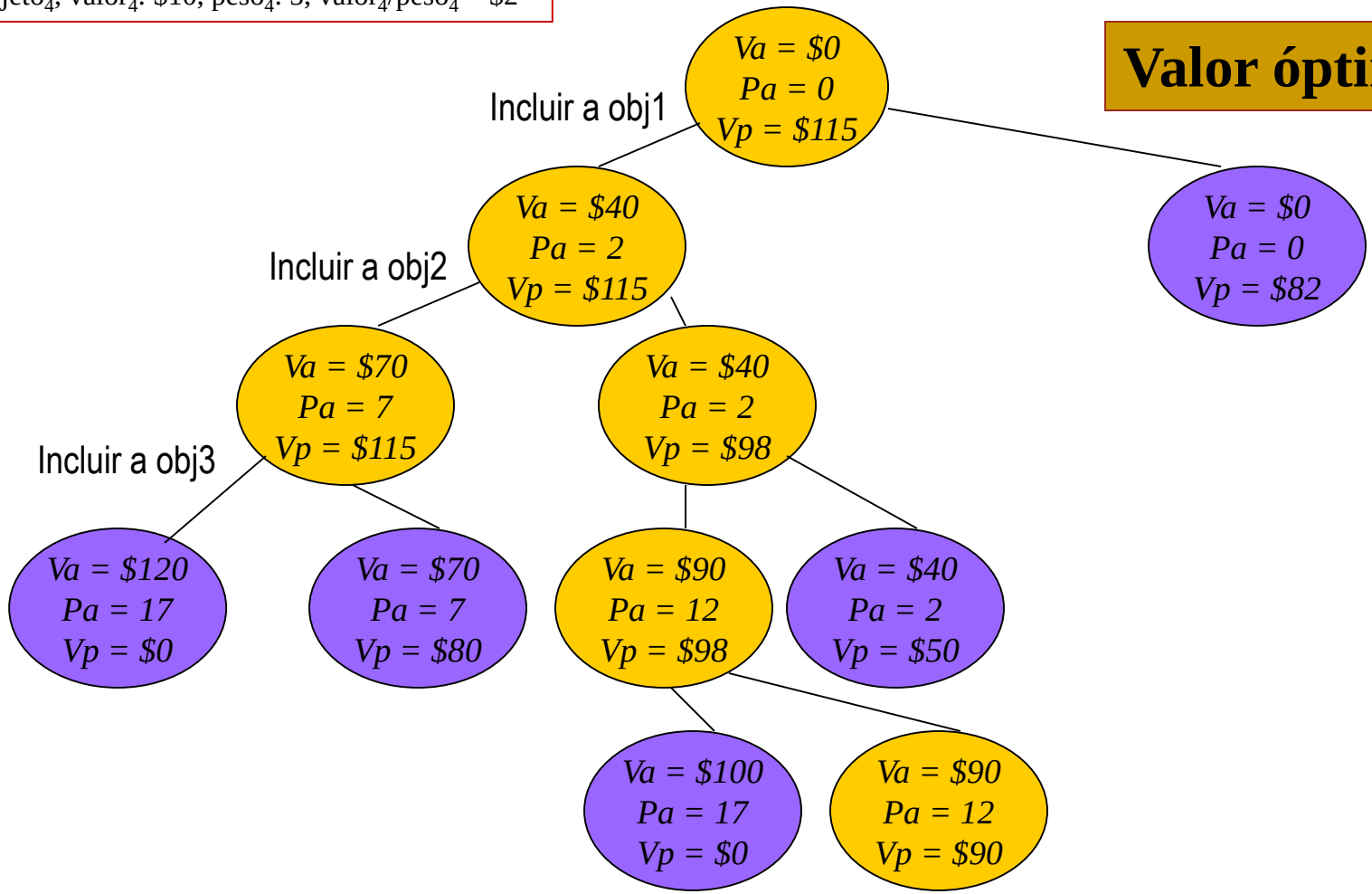
Valor óptimo = \$90

De los nodos a expandir,
¿cuál tiene el mejor valor posible?

Ejemplo

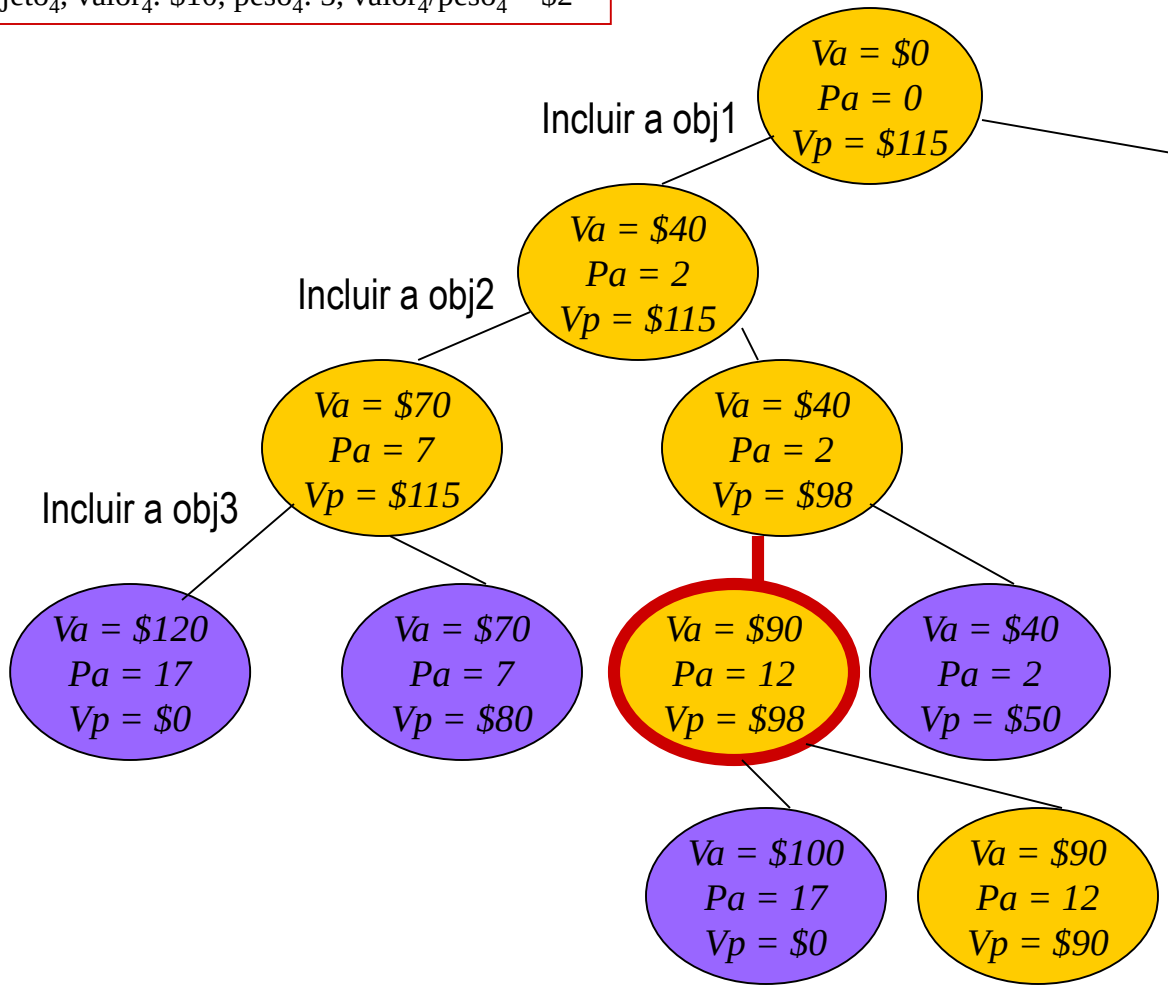
PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2

Valor óptimo = \$90



Ejemplo

PESO MOCHILA = 16
Objeto$_1$, valor$_1$: \$40, peso$_1$: 2, valor$_1$/peso$_1$ = \$20
Objeto$_2$, valor$_2$: \$30, peso$_2$: 5, valor$_2$/peso$_2$ = \$6
Objeto$_3$, valor$_3$: \$50, peso$_3$: 10, valor$_3$/peso$_3$ = \$5
Objeto$_4$, valor$_4$: \$10, peso$_4$: 5, valor$_4$/peso$_4$ = \$2



Valor óptimo = \$90

Solución que requirió sólo 11 nodos

Implementación del algoritmo

- Tomando como base el algoritmo del recorrido nivel por nivel, se utilizará una **cola priorizada** en vez de una fila…
- La prioridad la tendrá el nodo con mayor valor posible a acumular…
- El algoritmo se adapta al contexto del problema, considerando la formación de nodos, y su inserción y eliminación de la cola priorizada.
- Los nodos sólo guardan los siguientes valores de información: nivel, valor acumulado, peso acumulado y valor posible a acumular… (no hay necesidad de tener apuntadores).

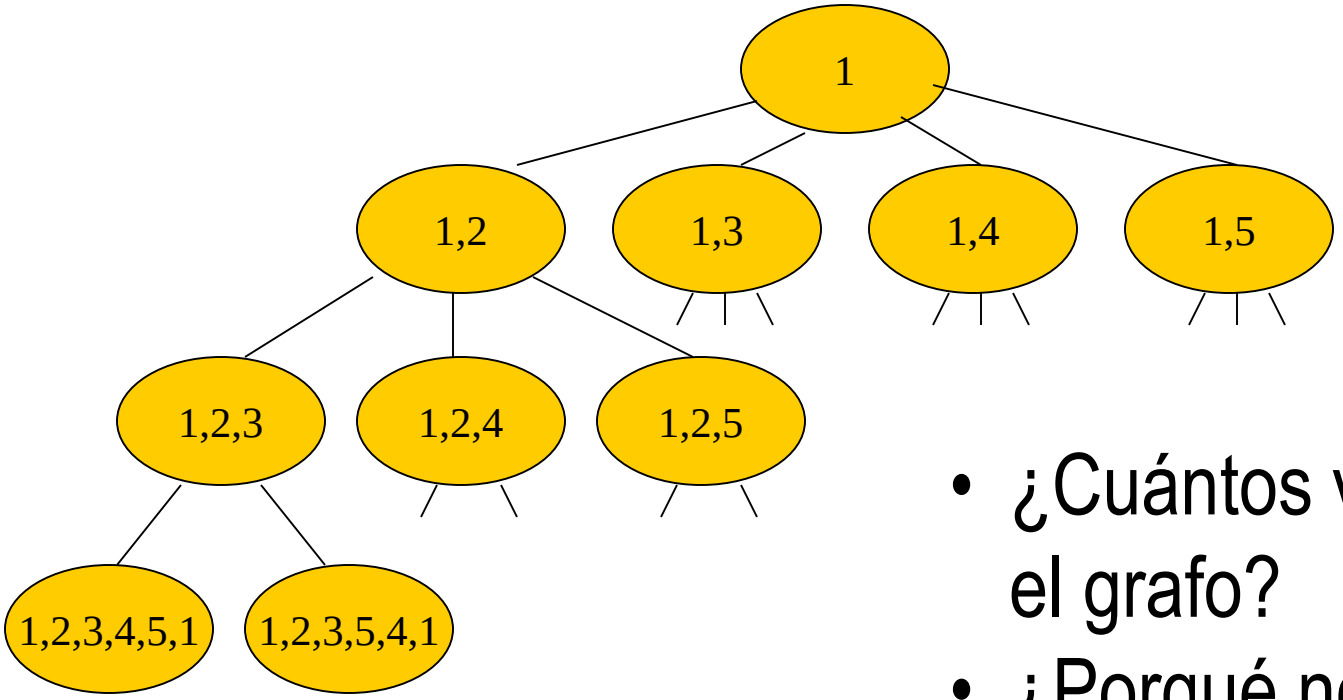
El problema del viajero

- Este problema fue resuelto con la Programación dinámica, obteniendo un algoritmo de orden **$O(n^2 2^n)$**…
- Para una ‘n’ grande, el algoritmo es ineficiente…
- Con Backtracking, se resolvió el problema de los ciclos Hamiltonianos, que indirectamente, resuelve el problema del viajero…
- Sin embargo, también puede llegar a tener un comportamiento exponencial o peor…
- Branch and bound se adapta para solucionarlo…

El problema del viajero

- *Recordando:* Ciclo en el grafo en el que TODOS los vértices del grafo se visitan sólo una vez al menor costo.
- Arbol de búsqueda de soluciones:
  - La raíz del árbol (nivel 0) es el vértice de inicio del ciclo.
  - En el nivel 1 se consideran TODOS los vértices menos el inicial.
  - En el nivel 2 se consideran TODOS los vértices menos los 2 que ya fueron visitados.
  - Y así sucesivamente hasta el nivel 'n-1' que incluirá al vértice que no ha sido visitado.

Ejemplo



- ¿Cuántos vértices tiene el grafo?
- ¿Porqué no se requiere el último nivel en el árbol?

Análisis del problema con Branch and bound

- Criterio de selección para expandir un nodo del árbol de búsqueda de soluciones:
  - Un vértice en el nivel i del árbol, debe ser adyacente al vértice en el nivel i-1 del camino correspondiente en el árbol.
  - Puesto que es un problema de MINIMIZACIÓN, si el costo posible a acumular al expandir el nodo i, es **menor** al mejor costo acumulado hasta ese momento, vale la pena expandir el nodo, si no, el camino ahi se deja de explorar...

Estimación del costo posible a acumular

- Si se sabe cuáles son los vértices que faltan por visitar…
- Cada vértice faltante, tiene arcos de salida hacia otros vértices…
- El mejor costo, será el del arco que tenga el valor menor…
- Esta información se puede obtener del renglón correspondiente al vértice en la matriz de adyacencias (excluyendo a los valores de cero)...
- La sumatoria de los mejores arcos de cada vértice faltante, más el costo del camino ya acumulado, es un estimado válido para tomar decisiones respecto a las podas en el árbol..

Ejemplo

- Dada la siguiente matriz de adyacencias, ¿cuál es el costo mínimo posible de visitar todos los nodos una sola vez?

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0

Mínimo = ***4***

Mínimo = ***7***

Mínimo = ***4***

Mínimo = ***2***

Mínimo = ***4***

TOTAL = 21

Ejemplo

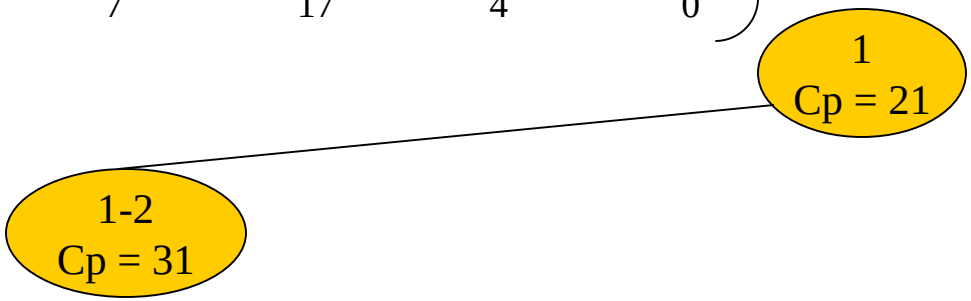
0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0

1
Cp = 21

Costo mínimo = ∞

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



Costo mínimo = ∞

Cálculo del Costo posible:

Acumulado de 1-2 : **14**

Más mínimo de 2-3, 2-4 y 2-5: **7**

Más mínimo de 3-1, 3-4 y 3-5: **4**

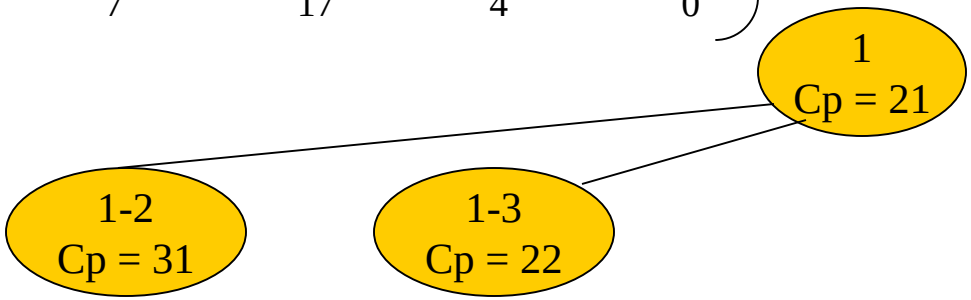
Más mínimo de 4-1, 4-3 y 4-5: **2**

Más mínimo de 5-1, 5-3 y 5-4: **4**

TOTAL = 31

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



Costo mínimo = ∞

Cálculo del Costo posible:

Acumulado de 1-3 : **4**

Más mínimo de 3-2, 3-4 y 3-5: **5**

Más mínimo de 2-1, 2-4 y 2-5: **7**

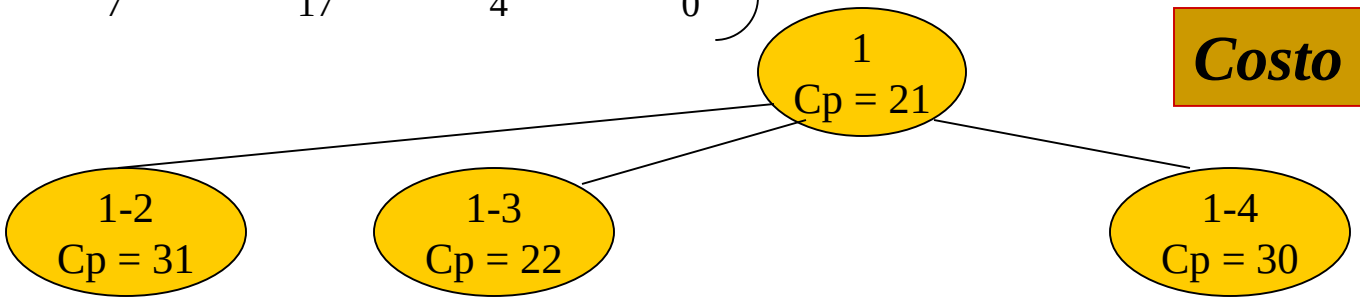
Más mínimo de 4-1, 4-2 y 4-5: **2**

Más mínimo de 5-1, 5-2 y 5-4: **4**

TOTAL = 22

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



Costo mínimo = ∞

Cálculo del Costo posible:

Acumulado de 1-4 : **10**

Más mínimo de 4-2, 4-3 y 4-5: **2**

Más mínimo de 3-1, 3-2 y 3-5: **4**

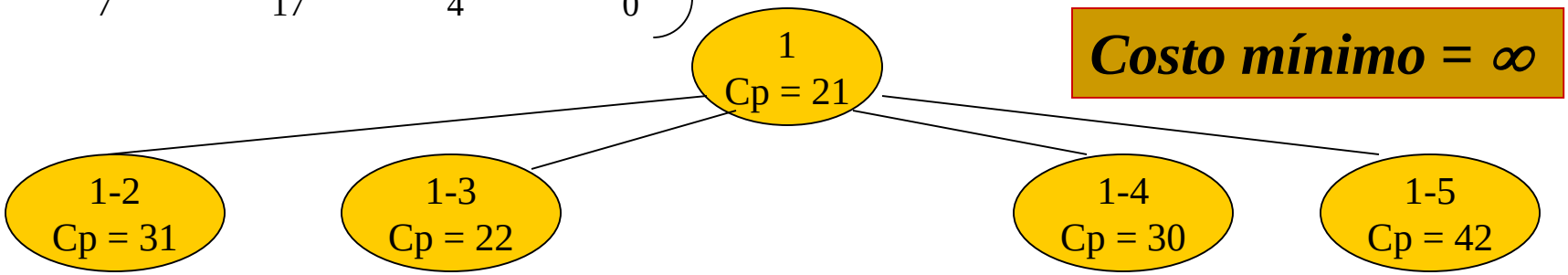
Más mínimo de 2-1, 2-3 y 2-5: **7**

Más mínimo de 5-1, 5-2 y 5-3: **7**

TOTAL = 30

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



Costo mínimo = ∞

¿Cuál es el mejor nodo para expandir?

Cálculo del Costo posible:

Acumulado de 1-5 : **20**

Más mínimo de 5-2, 5-3 y 5-4: **4**

Más mínimo de 4-1, 4-2 y 4-3: **7**

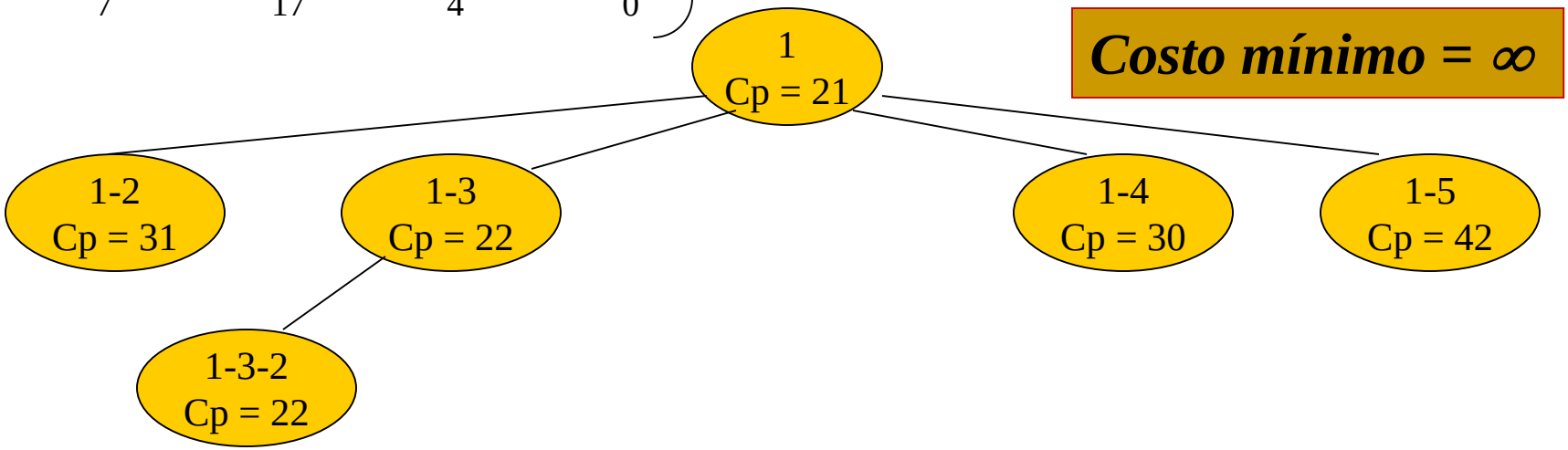
Más mínimo de 3-1, 3-2 y 3-4: **4**

Más mínimo de 2-1, 2-3 y 2-4: **7**

TOTAL = 42

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



Costo mínimo = ∞

Cálculo del Costo posible:

Acumulado de 1-3-2 : **9**

Más mínimo de 2-4 y 2-5: **7**

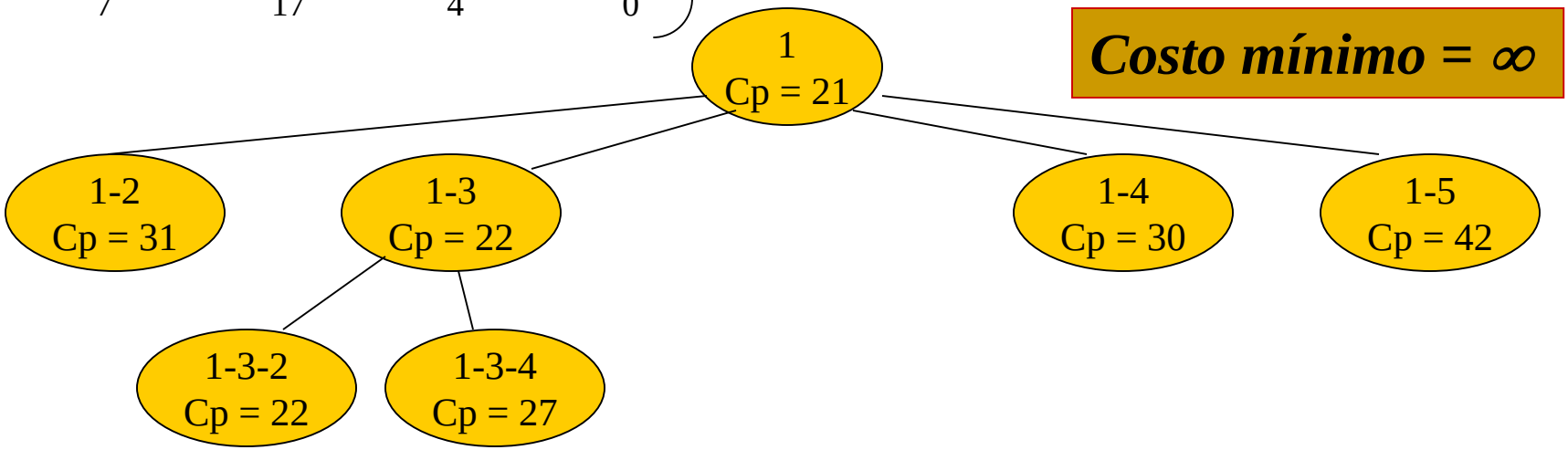
Más mínimo de 4-1 y 4-5: **2**

Más mínimo de 5-1 y 5-4: **4**

TOTAL = 22

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



Costo mínimo = ∞

Cálculo del Costo posible:

Acumulado de 1-3-4 : **11**

Más mínimo de 4-2 y 4-5: **2**

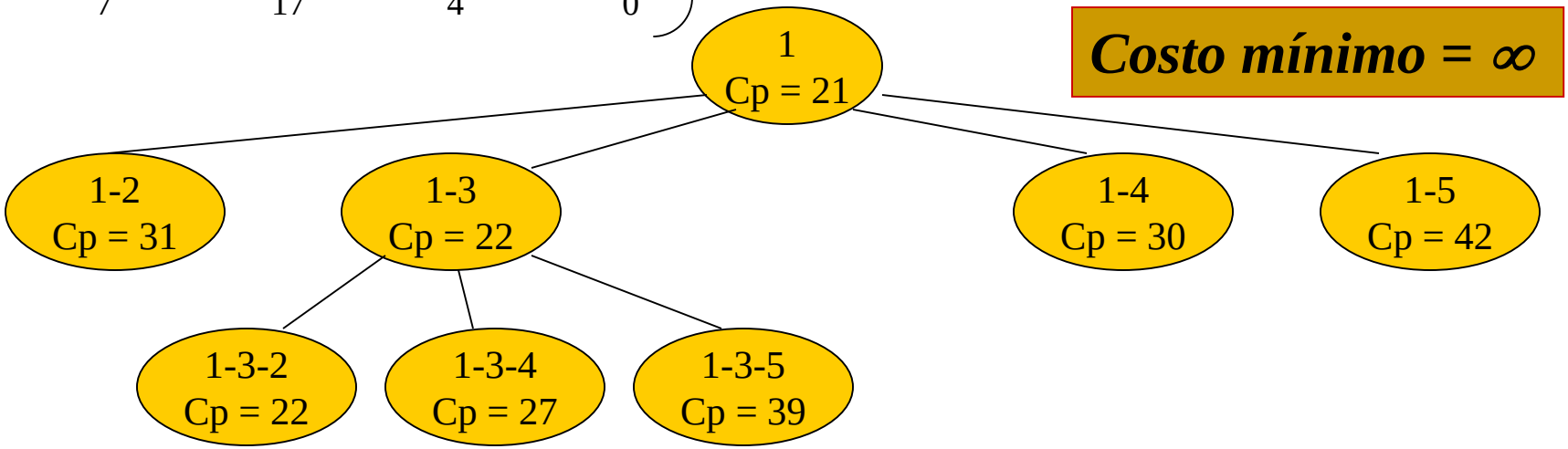
Más mínimo de 2-1 y 2-5: **7**

Más mínimo de 5-1 y 5-2: **7**

TOTAL = 27

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



Costo mínimo = ∞

¿Cuál es el mejor nodo para expandir?

Cálculo del Costo posible:

Acumulado de 1-3-5 : **20**

Más mínimo de 5-2 y 5-4: **4**

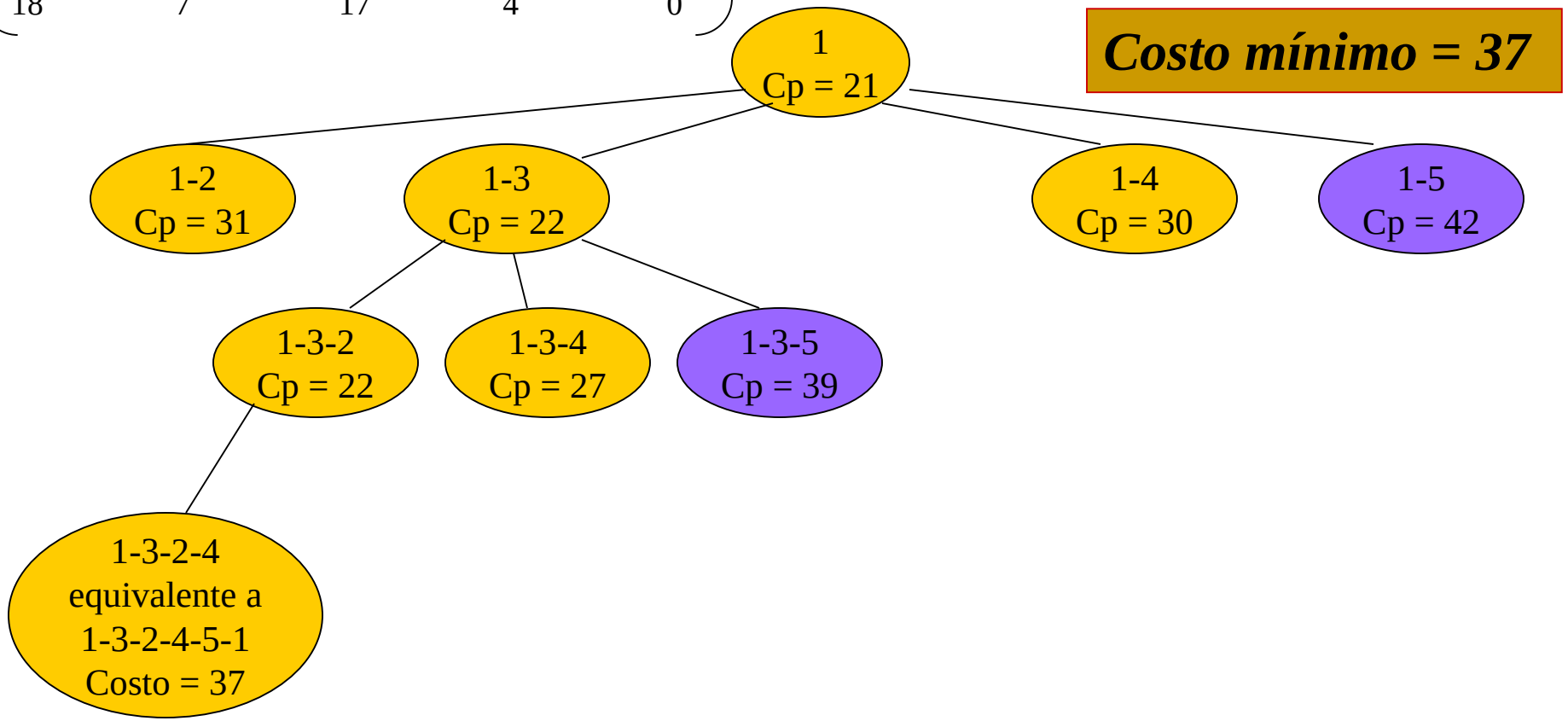
Más mínimo de 2-1 y 2-4: **8**

Más mínimo de 4-1 y 4-2: **7**

TOTAL = 39

Ejemplo

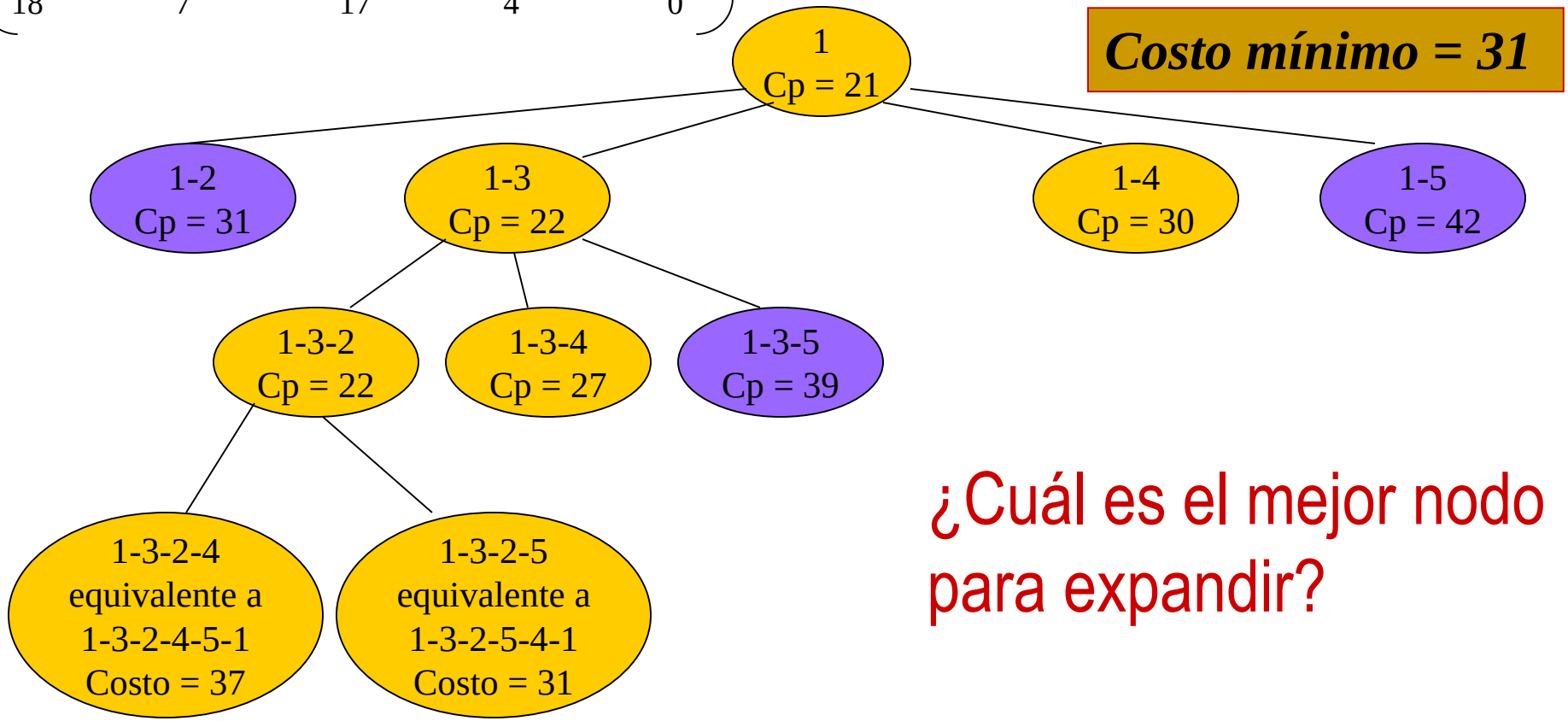
0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



Costo mínimo = 37

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0

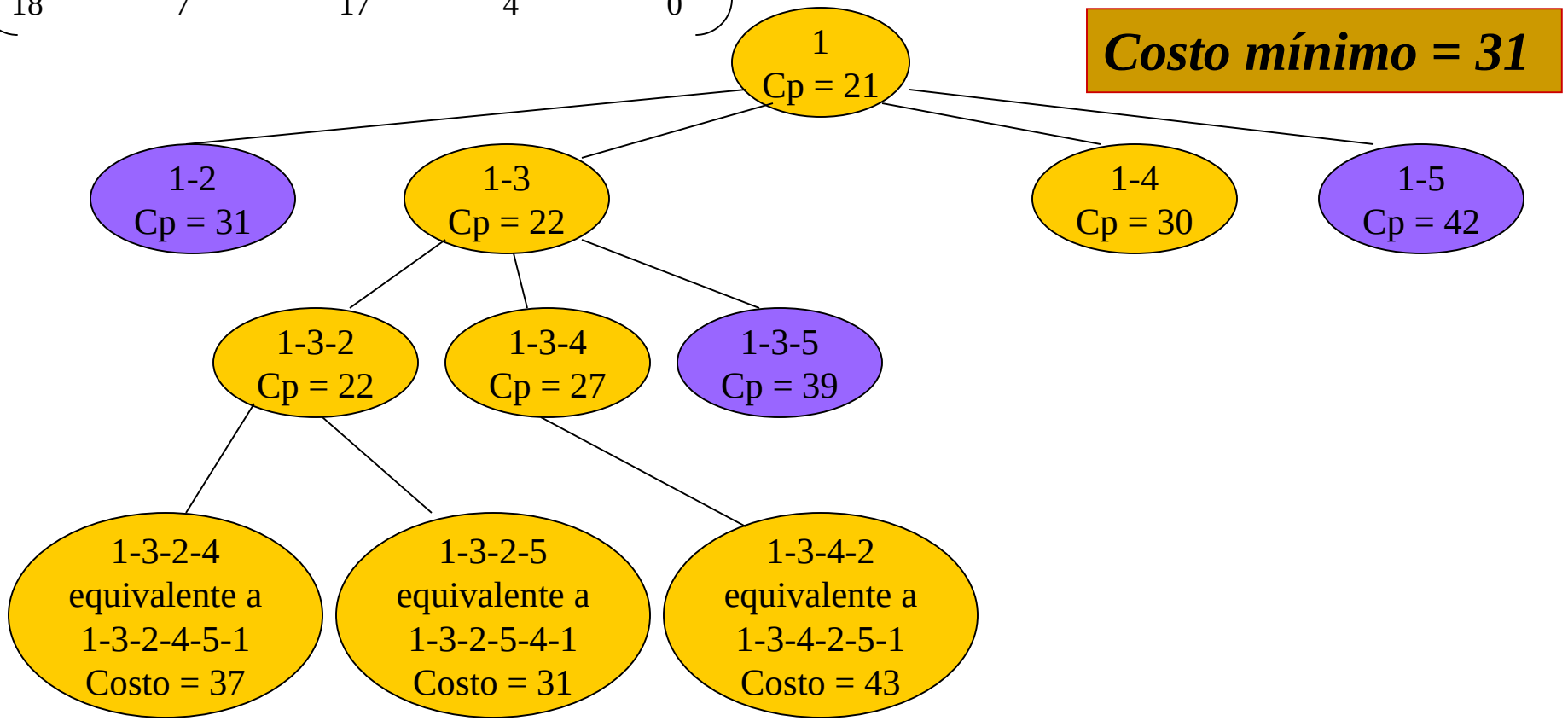


Costo mínimo = 31

¿Cuál es el mejor nodo para expandir?

Ejemplo

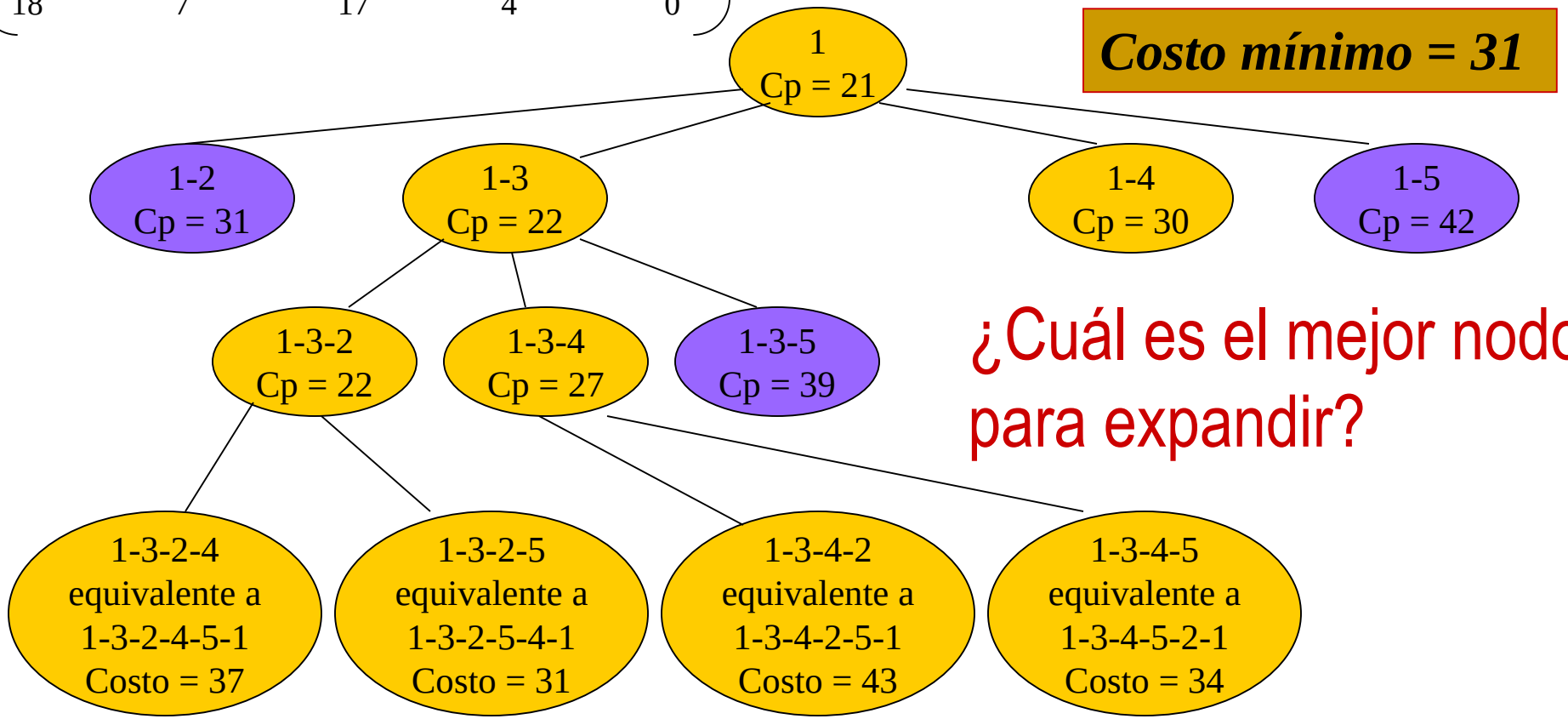
0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



Costo mínimo = 31

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0



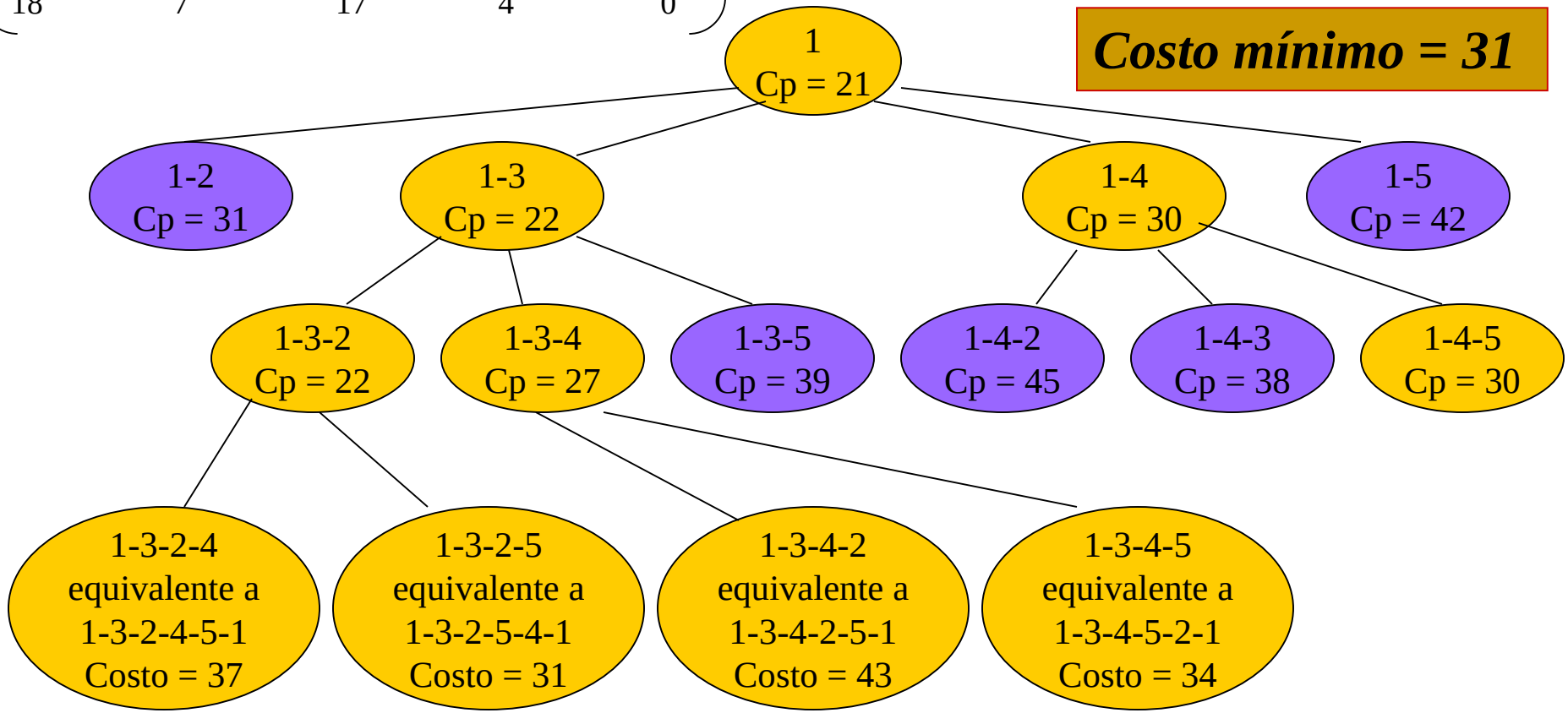
Costo mínimo = 31

¿Cuál es el mejor nodo para expandir?

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0

Costo mínimo = 31

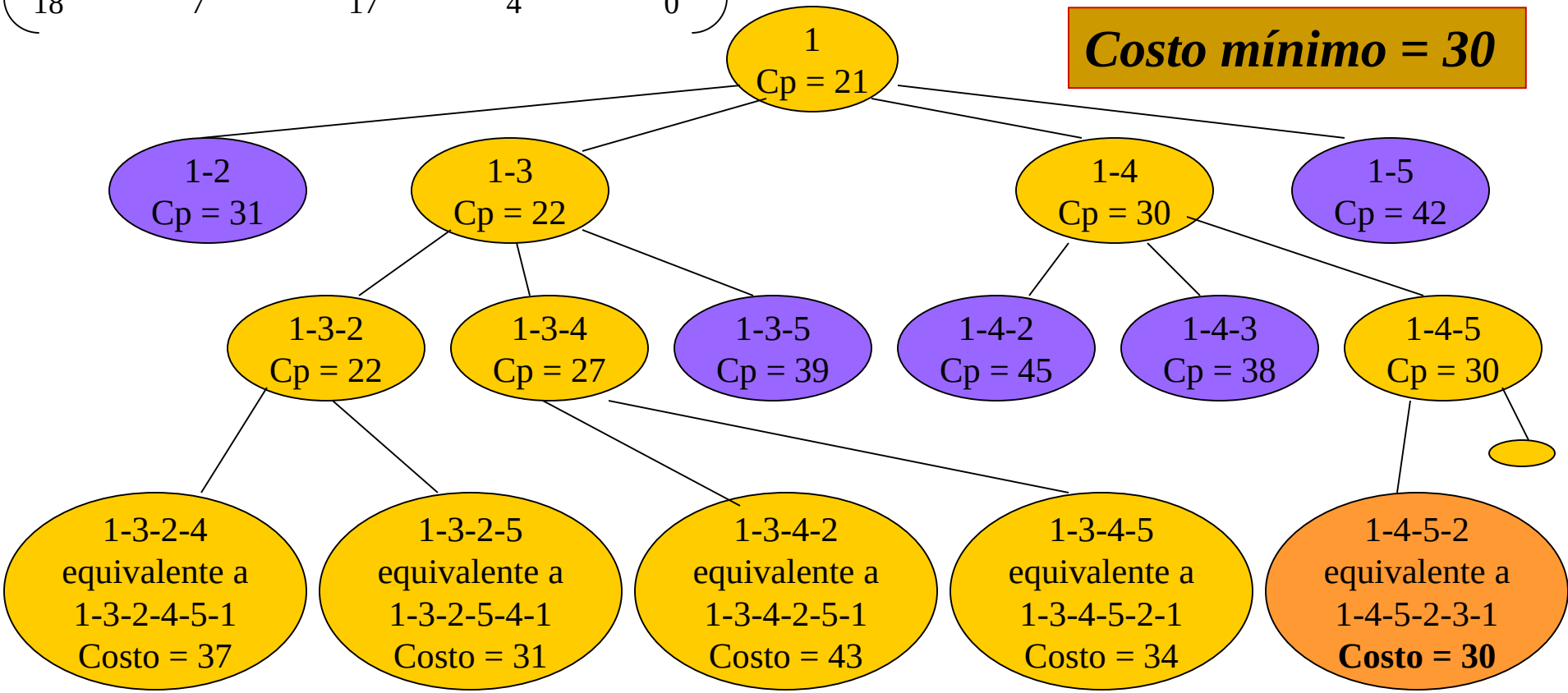


¿Cuál es el mejor nodo para expandir?

Ejemplo

0	14	4	10	20
14	0	7	8	7
4	5	0	7	16
11	7	9	0	2
18	7	17	4	0

Costo mínimo = 30



Conclusión final: El problema del viajero

- Branch and bound ofrece una opción más a la solución del problema del viajero…
- Sin embargo, NO asegura tener un buen comportamiento en cuanto a eficiencia, pues en el peor caso, tiene un comportamiento exponencial…
- El problema puede ser resuelto con la técnica de los “algoritmos aproximados”...

CASO

- Se desea asignar a ***n*** personas a ***n*** trabajos distintos. La asignación de cierta persona a cierto trabajo tiene un costo, es decir, la asignación de la persona *i* al trabajo *j* tiene el costo C_{ij} . ¿Cuál es la asignación de personas que minimizaría el costo total de asignaciones?