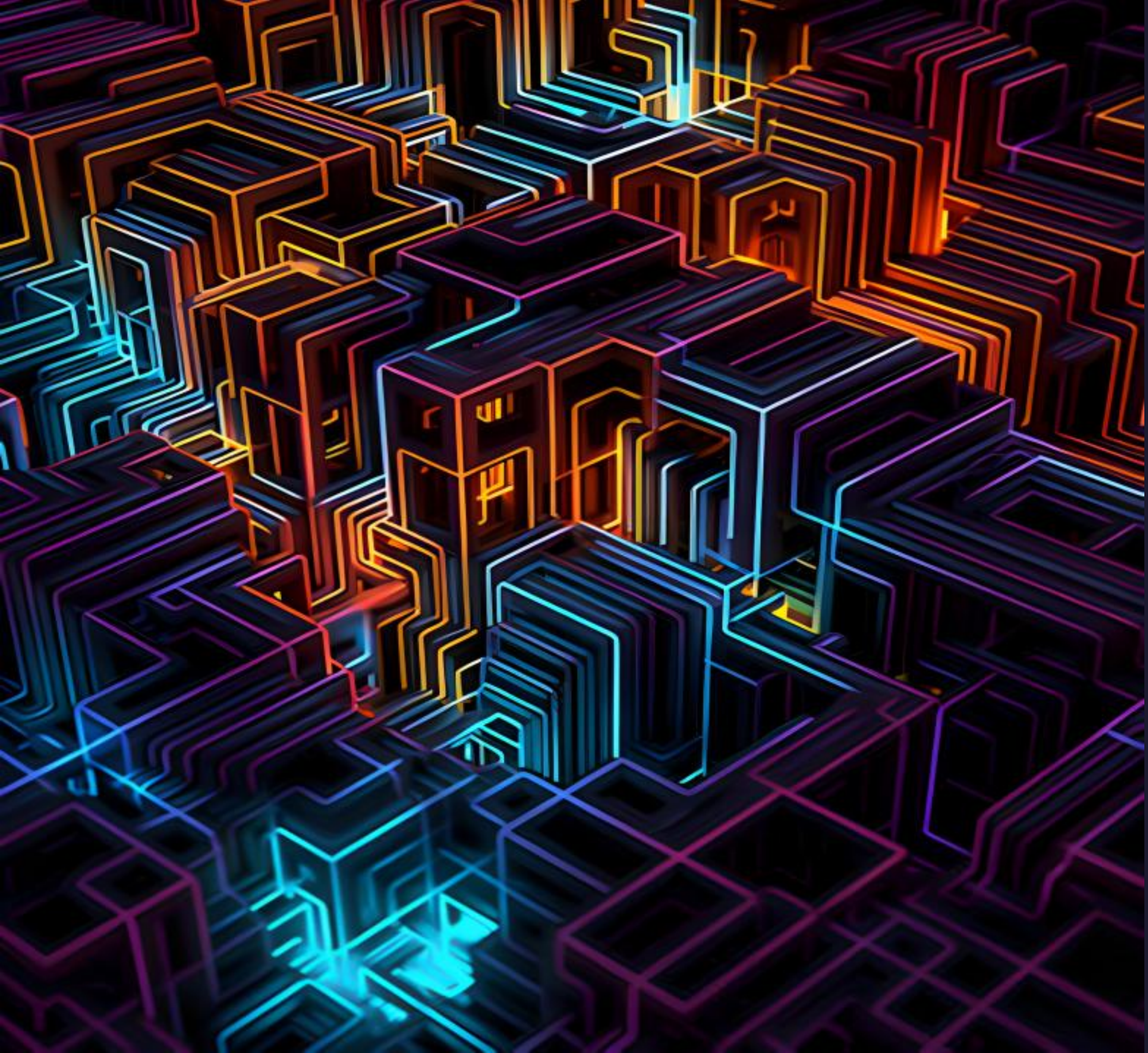


Técnicas de diseño de algoritmos: branch-and-bound

Análisis y diseño de algoritmos avanzados

Dra. Valentina Narváez Terán

Tecnológico de Monterrey

La perspectiva (mas o menos) completa

Cuando tratamos con problemas de optimización, hay dos grandes ramas de algoritmos

- Algoritmos
  - **Exactos**
    Óptimos asegurados
    - Búsqueda exhaustiva
    - Backtraking B&B
    - Solvers declarativos (como CPLEX)
  - **Aproximados**
    Óptimos no asegurados
    - **Heurísticas**
      Intuiciones educadas sobre el problema, para construir soluciones
      - Algs. Greedy
      - Programación dinámica
      - Randomized
    - **Metaheurísticas**
      Heurísticas sobre heurísticas: Mejorar soluciones existentes, vía modificación y/o combinación
      - Monosolución
      - Poblacionales
    - **Hiperheurísticas**
      Optimización de metaheurísticas y sus componentes

En estas ramas, la investigación se centra en: algoritmos para problemas nuevos, algoritmos mejores que los existentes, estudios empíricos de dificultad… etc., etc., etc.

…

Backtraking vs branch-and-bound

B&B:
Una técnica muy parecida a backtracking, pero para problemas de optimización, en lugar de satisfacción de restricciones

También conocido como ramificación-y-poda

Satisfacción de restricciones vs optimización: la diferencia es ¿qué buscamos?
Soluciones factibles vs soluciones optimas

Solución factible: asignación de variables que cumple las restricciones

Solución optima: asignación de variables que minimiza (o maximiza) una función de costo

Backtraking vs branch-and-bound

¿En que se parecen?

- Podemos imaginar la exploración del espacio de soluciones como un árbol
- Partimos de una solución raíz indefinida
- Una fila para organizar el proceso
- Las soluciones salen de la fila, y se ramifican en nodos hijos que entran en la fila si son potencialmente factibles

¿En que son diferentes?

- No basta una solución factible (que satisfaga las restricciones), buscamos óptimos (que minimice o maximice una función objetivo)
- Fila de prioridad. Explorar primero las soluciones parciales de mejor costo estimado
- Descartamos soluciones hijas invalidas, y además las que tengan peores valores de la función objetivo

Técnicas de diseño: branch and bound

Revisitando…

Knapsack Problem

Un knapsack de capacidad K, y conjunto de objetos con valor $V = \{v_1, v_2, \dots, v_n\}$, y sus correspondientes pesos $W = \{w_1, w_2, \dots, w_n\}$, elegir un subconjunto de objetos S

Maximizar

$$\sum_{i=1}^{|S|} v_i$$

Esto es la función objetivo del problema

Buscamos que el valor total de los objetos elegidos sea lo mayor posible

Sujeto a:

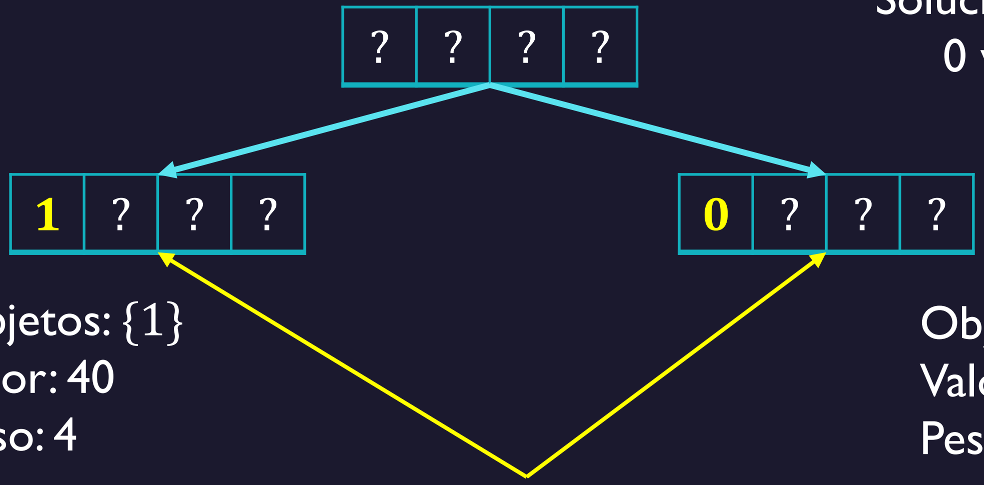
$$\sum_{i=1}^{|S|} w_i \leq K$$

Y sus restricciones

Que pesos no sobrepasen la capacidad del knapsack

Técnicas de diseño: branch-and-bound

Solución indefinida con
0 valor y 0 peso



Objetos: $\{1\}$
Valor: 40
Peso: 4

Objetos: $\{\emptyset\}$
Valor: 0
Peso: 0

Soluciones parcialmente definidas

Instancia de knapsack

objeto	peso w_i	valor v_i	ratio $\frac{v_i}{w_i}$
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4

Capacidad $K = 10$

¿Cual es la mejor solución parcial?
La que resulte serlo, será la rama *mas prometedora*

Al entrar en la fila, tendrá prioridad para salir antes que otras soluciones parciales menos prometedoras, así que sus ramificaciones se exploraran primero

Técnicas de diseño: branch-and-bound

¿Cómo decidimos cual rama es mas prometedora?

1	?	?	?

0	?	?	?

La que tenga la mejor suma de:

- El costo parcial de la solución
- Una estimación del mejor valor posible dados sus elementos indefinidos.
  Se diseña según el problema y puede ser mas o menos certera.

Esto produce una cota superior o inferior del mejor costo (si el problema es de maximización o de minimización) en toda la rama a partir de la solución parcial

En knapsack, la cota es hipotéticamente, el máximo valor de la mejor solución que podría existir en la rama derivada de una solución padre

Técnicas de diseño: branch-and-bound

¿Cuál es la cota superior de la solución inicial?

Instancia de knapsack

objeto	peso w_i	valor v_i	ratio $\frac{v_i}{w_i}$
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4

Capacidad $K = 10$

Si la solución parcial ha acumulado un valor de v y un peso de w, en el knapsack queda $(K - w)$ de capacidad libre.

Si cada unidad de capacidad libre fuese ocupada por el mejor ratio, la mejor solución podría tener un hipotético valor tope de:

$$ub = v + \underbrace{(K - w)}_{\text{unidades de capacidad libres}} \underbrace{(best\ ratio)}_{\text{mejor ratio de entre los objetos pendientes de elegir (variables aun indefinidas)}}$$

valor de los objetos elegidos

Técnicas de diseño: branch-and-bound

$$ub = \underset{\text{valor de los elegidos (variables definidas)}}{v} + \underbrace{(K - w)}_{\text{unidades de capacidad libres}} \underbrace{(best\ ratio)}_{\text{mejor ratio de los objetos pendientes de elegir (variables aun indefinidas)}}$$

objeto	peso w_i	valor v_i	ratio $\frac{v_i}{w_i}$
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4

Capacidad $K = 10$

Entonces, para la solución raíz y sus hijas, las estimaciones son:

| ? | ? | ? | ? |

$0 + (10 - 0)(10) = 100$

| **1** | ? | ? | ? |

$40 + (10 - 4)(6) = 76$

| **0** | ? | ? | ? |

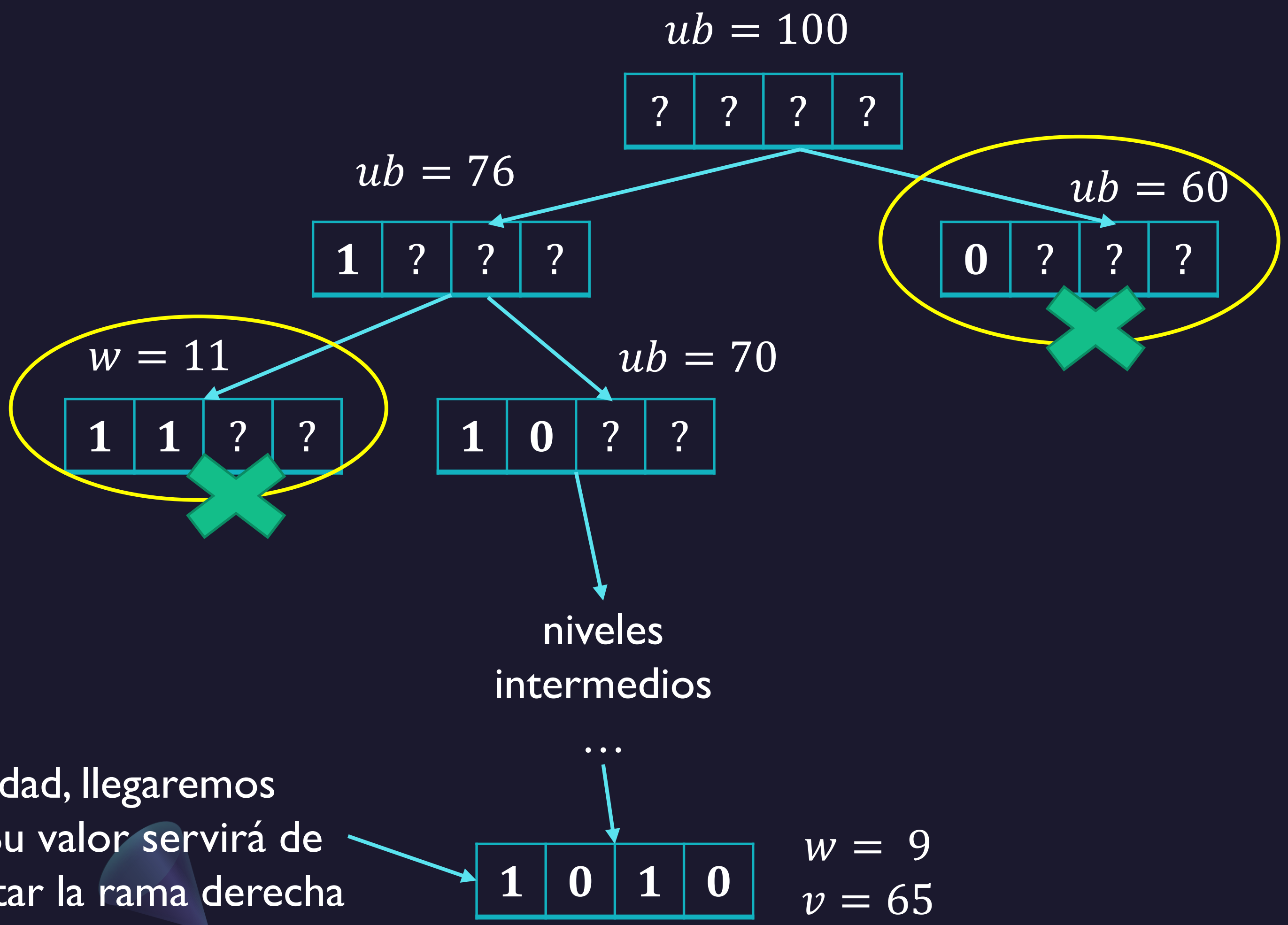
$0 + (10 - 0)(6) = 60$

Esta solución entrará en la fila con mayor prioridad, causando que salga antes, y su rama sea explorada primero

Técnicas de diseño: branch-and-bound

¿Cómo podamos ramas?

- Si la solución hija es invalida (su peso sobrepasa K) no entra en la fila
- Si al salir de la fila, su cota (ub) es peor que el costo de la mejor solución completamente definida encontrada hasta el momento, no tendrá hijas



Por la fila de prioridad, llegaremos primero a esta hoja. Su valor servirá de referencia para descartar la rama derecha

Técnicas de diseño: branch-and-bound

El algoritmo general de B&B:

1. La solución raíz entra en la fila
2. $bestSoFar =$ es la mejor solución completa hasta el momento

3. Mientras la fila no este vacía:
   4. $partial\_sol =$ solución que sale de la fila
   5. $i\ =$ siguiente variable indefinida de $partial\_sol$

   6. Por cada valor j en el dominio de i:
      7. Crea una solución hija, dándole a la variable i el valor j
      8. Evalúa el $upperBound$ de la solución hija
      9. Si la solución hija esta completamente definida, es factible y mejor que bestSoFar, remplaza a bestSoFar

      10. Si la solución hija esta parcialmente definida, es factible y su $upperBound$ es mejor que el costo de bestSoFar, agregarla a la fila, con $upperBound$ como prioridad

Técnicas de diseño: branch-and-bound

El algoritmo general de B&B:

1. La solución raíz entra en la fila
2. $bestSoFar =$ es la mejor solución completa hasta el momento

3. Mientras la fila no este vacía:
    3. $partial\_sol =$ solución que sale de la fila
    4. $i\ =$ primer variable indefinida de $partial\_sol$

    5. Por cada valor j en el dominio de i:
        6. Crea una solución hija, dándole a la variable i el valor j
        7. Evalúa el $upperBound$ de la solución hija
        8. Si la solución hija esta completamente definida, es factible y mejor que bestSoFar, remplaza a bestSoFar

        9. Si la solución hija esta parcialmente definida, es factible y su $upperBound$ es mejor que el costo de bestSoFar, agregarla a la fila, con $upperBound$ como prioridad

Nota sobre filas de prioridad:

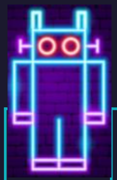
Generalmente, la prioridad es un número, y los valores menores tienen preferencia.

Si queremos darle preferencia a valores mayores de $upperBound$, la prioridad deberá ser $-upperBound$

Actividad 1.5: coin-collecting

Tienes una cuadricula de tamaño n. En las celdas marcadas hay monedas de oro

Un robot minero, que inicia en $[0,0]$, debe recolectarlas… pero esta averiado!
El robot solo se puede mover a la derecha o hacia abajo, nunca a izquierda o arriba.



i \ j	0	1	2	3	4
0		5			1
1	1				
2	7				1
3			2		
4	2		1	4	

Fechas: Dentro de una semana

Para crear tu algoritmo, es importante que definas:

- ¿Cuántas variables necesitas? ¿Qué dominio tienen?
  La i-th variable es el i-th movimiento. Cada movimiento es derecha o abajo
- ¿Cómo se ve la solución raíz? ¿Cómo crear soluciones hijas?
- ¿Cuál es la función objetivo?
  ¿Cómo calcular el upper-bound de una solución parciales?
  Recuerda que es una estimación de mejor caso