Algoritmos y Computabilidad

Estrategia de Ramificación y Acotación (Branch and Bound)

Modelo de la mochila

Maximizar $45x_1 + 48x_2 + 35x_3$

Dado, $5x_1 + 8x_2 + 3x_3 \leq 10$

$x_i \in \{0, 1\} \quad (i \in 1..3)$

¿Podríamos relajar algo más?

- Si los objetos fueran fraccionables

Maximizar $45x_1 + 48x_2 + 35x_3$

Dado, $5x_1 + 8x_2 + 3x_3 \leq 10$

$0 \leq x_i \leq 1 \quad (i \in 1..3)$

Relajación lineal

Modelo de la mochila

- ¿Qué implicación tiene la relajación lineal en el problema de la mochila?
  - Podemos ordenar en orden decreciente por la 'densidad' V_i / W_i

- ¿Cómo se resuelve ahora la relajación lineal?
  - Seleccionar ítems mientras quepan en la mochila
  - Seleccionar una fracción del último ítem
- En este ejemplo
  - $V_1 / W_1 = 9$, $V_2 / W_2 = 6$, $V_3 / W_3 = 11.7$
  - Orden 3,1,2
  - Seleccionamos ítems 3 y 1
  - Seleccionamos ¼ del ítem 2
  - Estimación: 92

$$45x_1 + 48x_2 + 35x_3$$

$$5x_1 + 8x_2 + 3x_3 \leq 10$$
$$0 \leq x_i \leq 1 \quad (i \in 1..3)$$

Ramificación y Acotación (en profundidad)

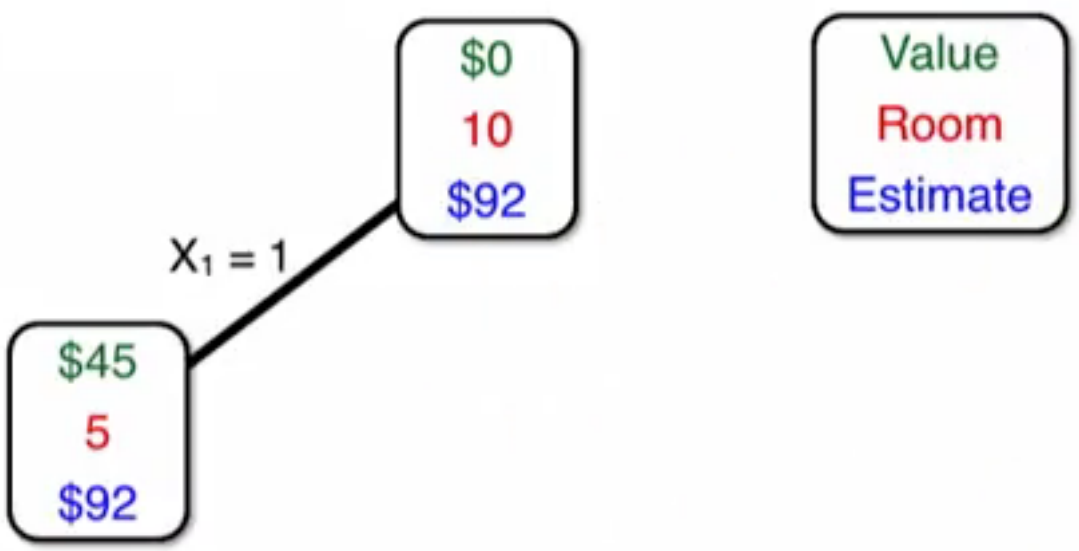
i	V_i	W_i
1	45	5
2	48	8
3	35	3

K = 10

Ramificación y Acotación (en profundidad)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

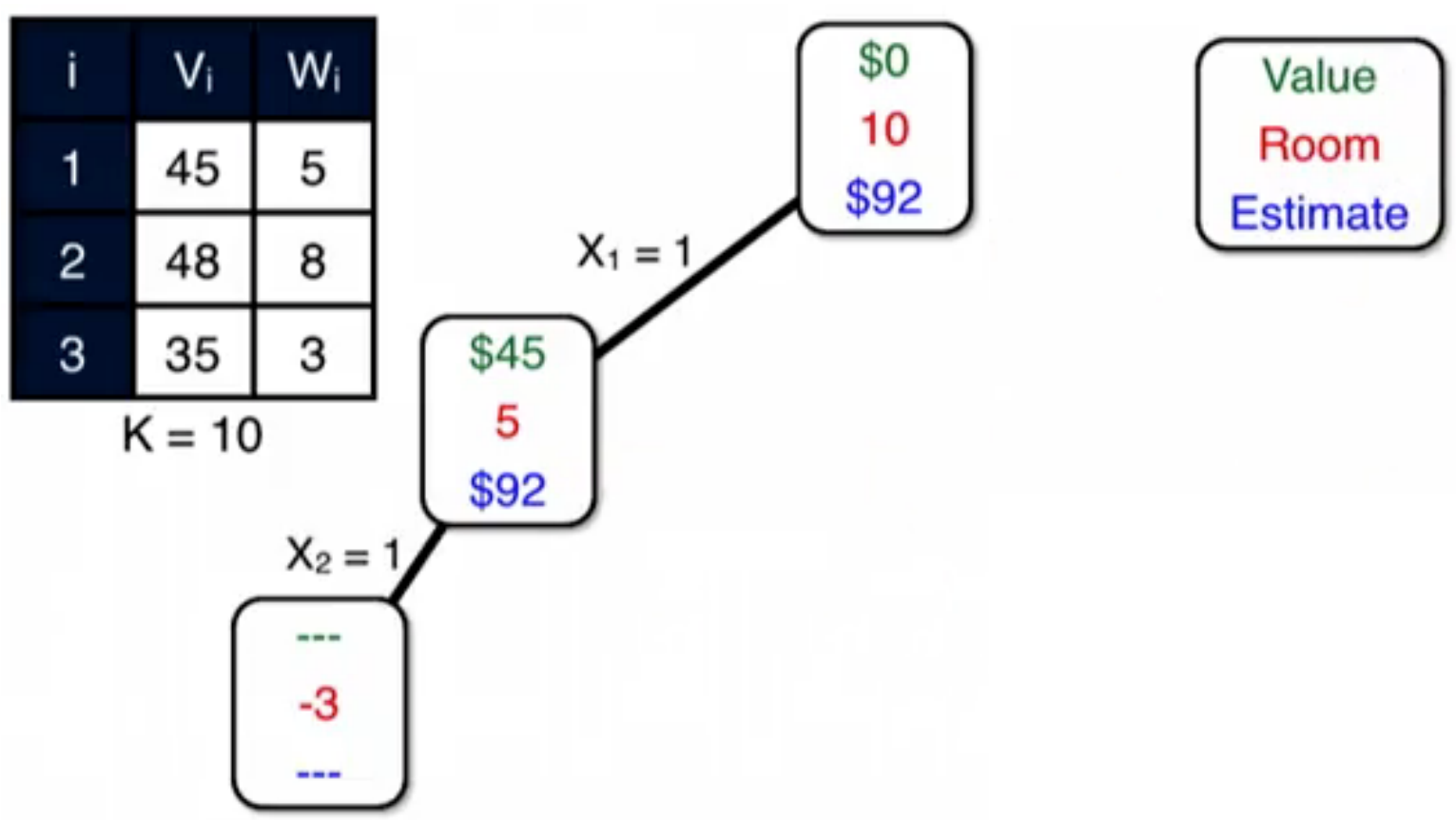
K = 10



Ramificación y Acotación (en profundidad)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

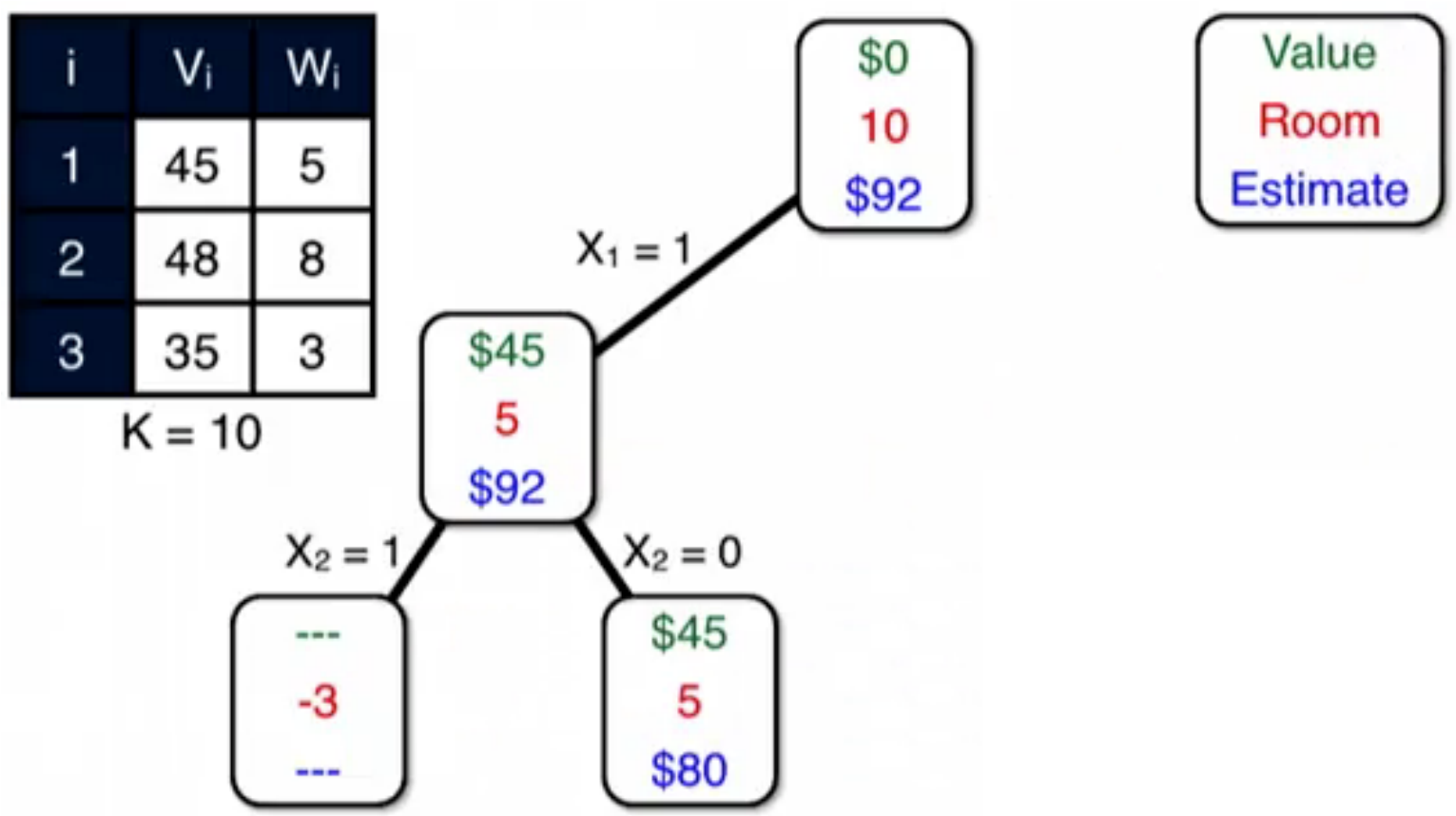
K = 10



Ramificación y Acotación (en profundidad)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

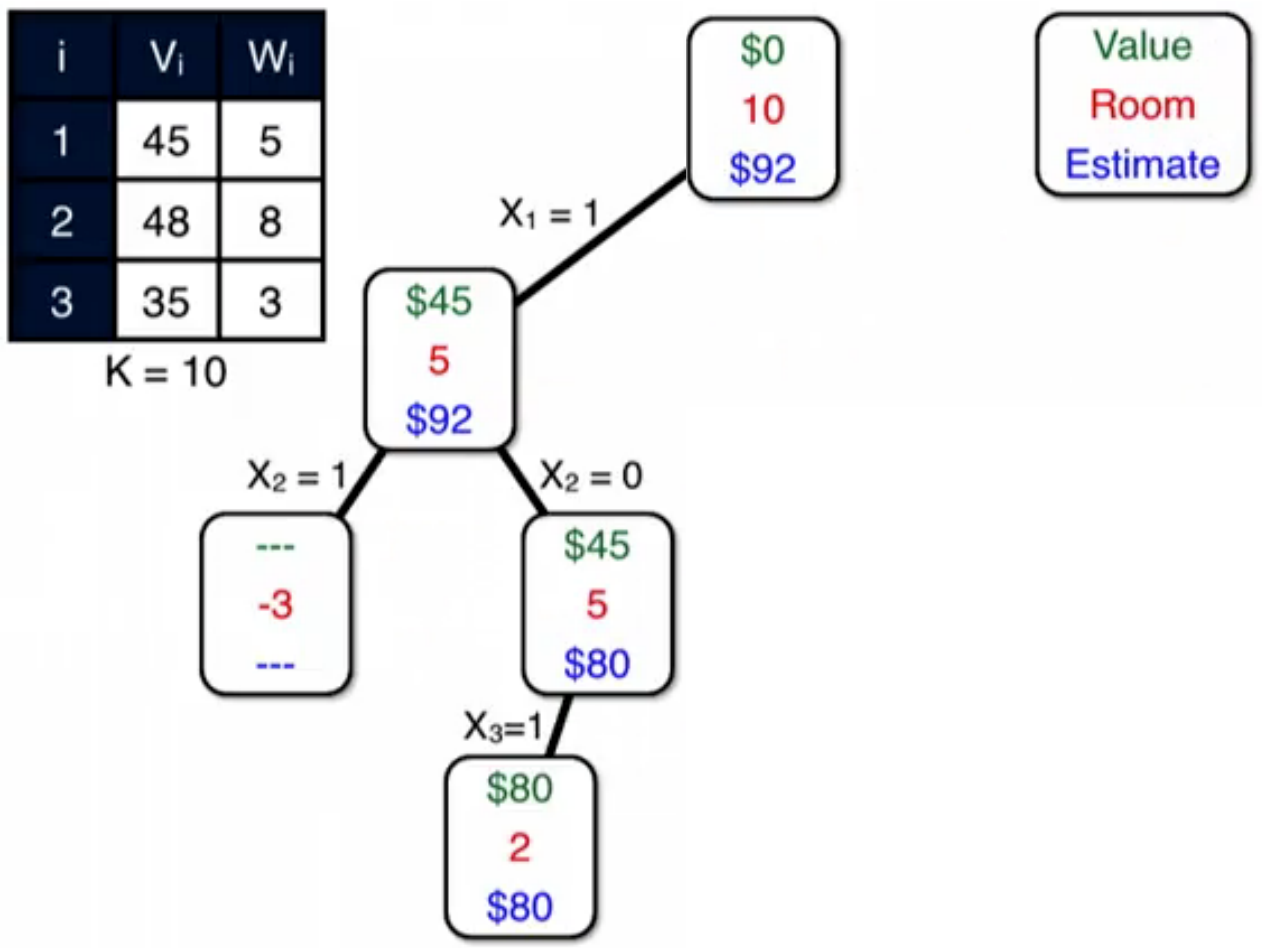
K = 10



Ramificación y Acotación (en profundidad)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

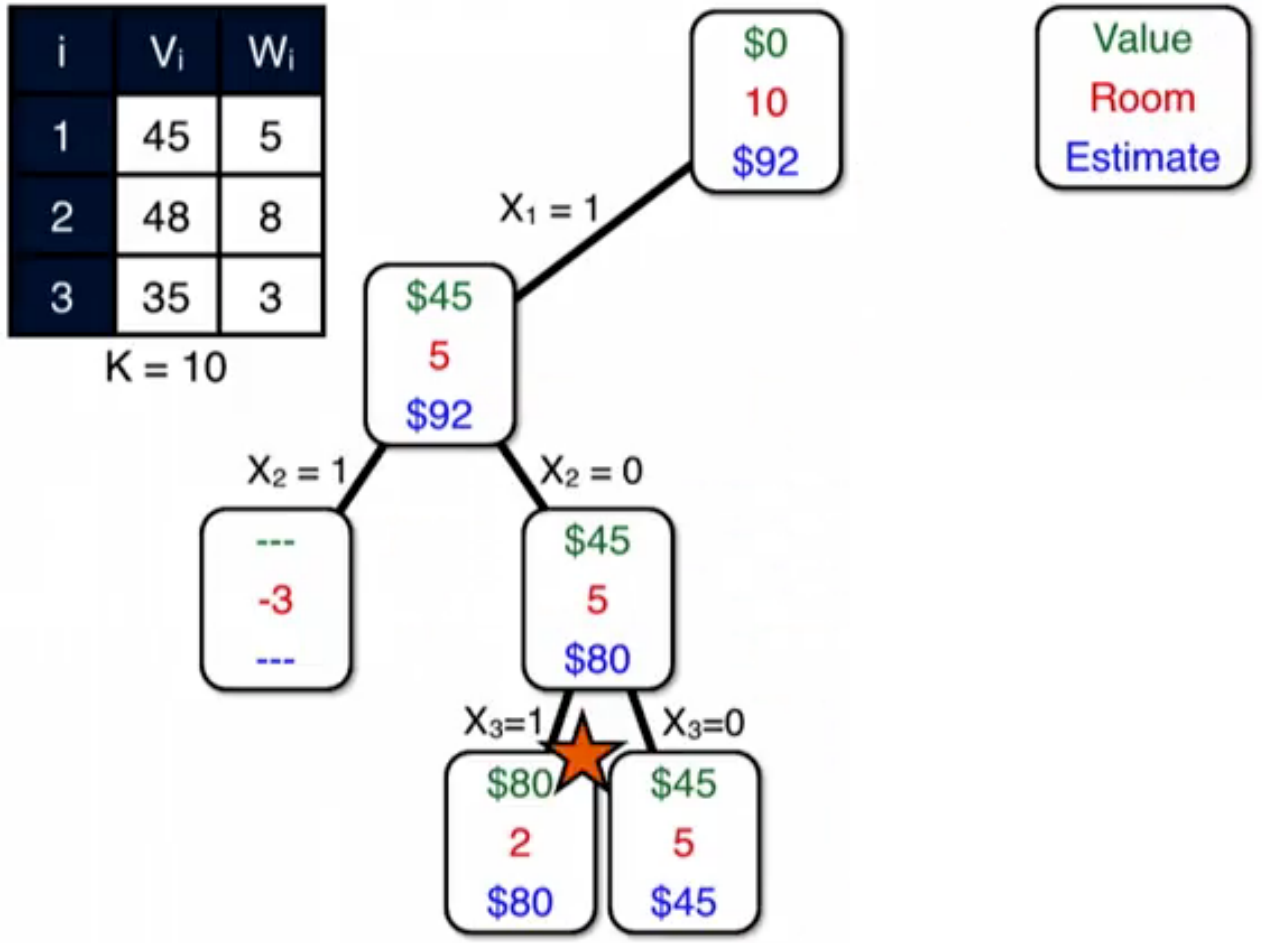
K = 10



Ramificación y Acotación (en profundidad)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

K = 10

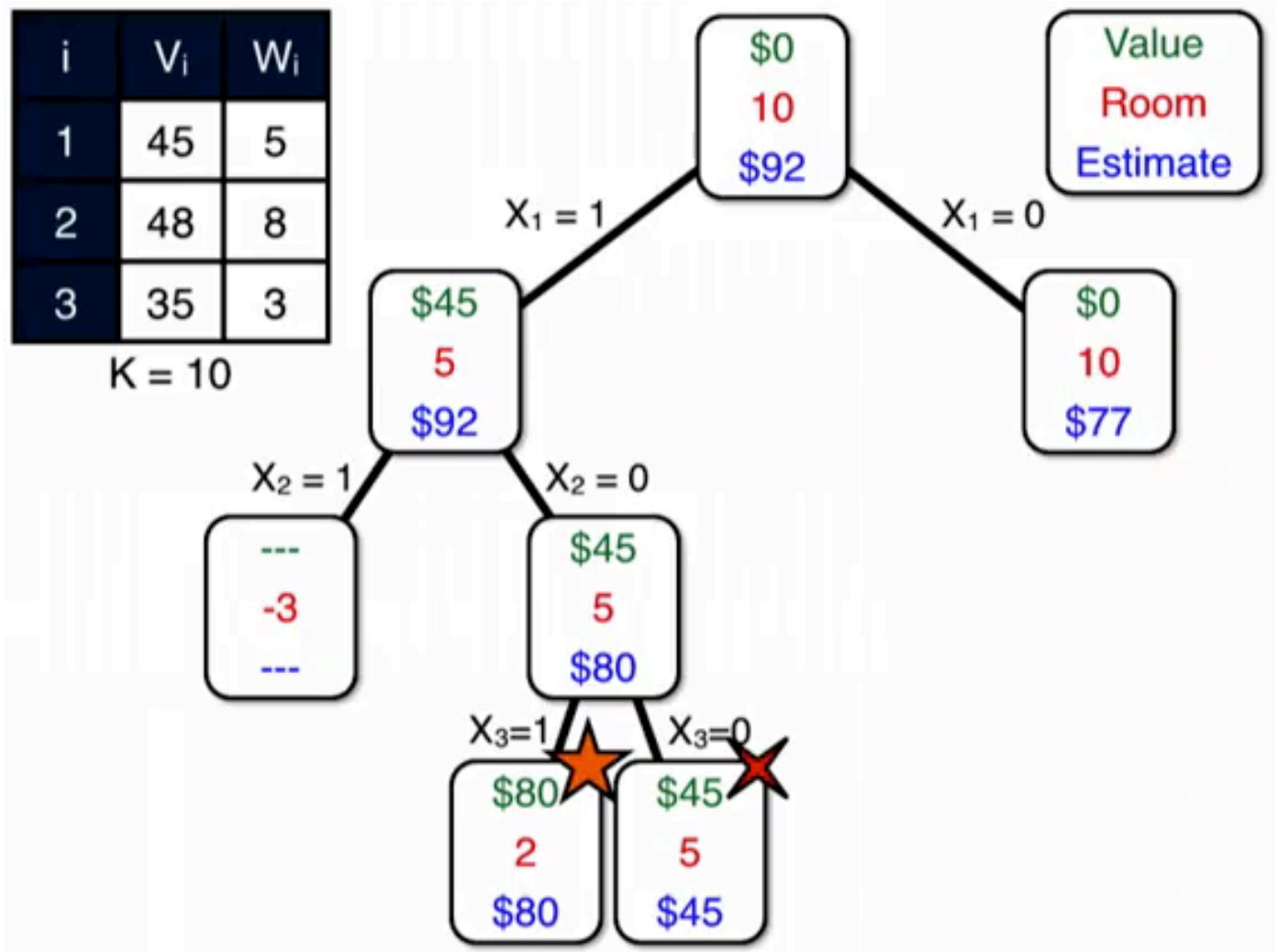


Ramificación y Acotación (en profundidad)

$V_1 / W_1 = 9$, $V_2 / W_2 = 6$, $V_3 / W_3 = 11.7$

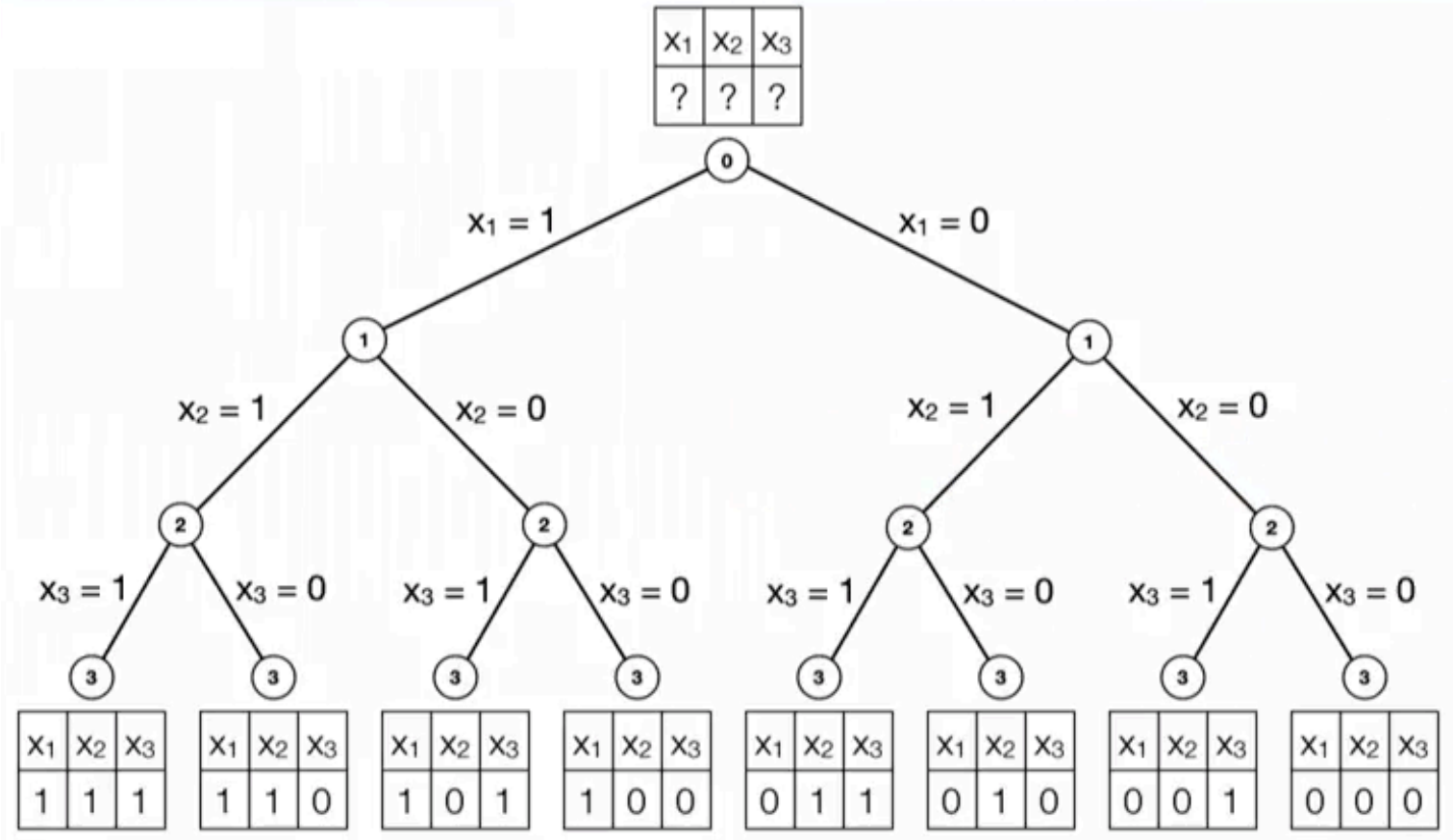
i	V_i	W_i
1	45	5
2	48	8
3	35	3

K = 10



- Objetivo
  - Introducir distintas estrategias de búsqueda para la ramificación y acotación

Búsqueda exhaustiva



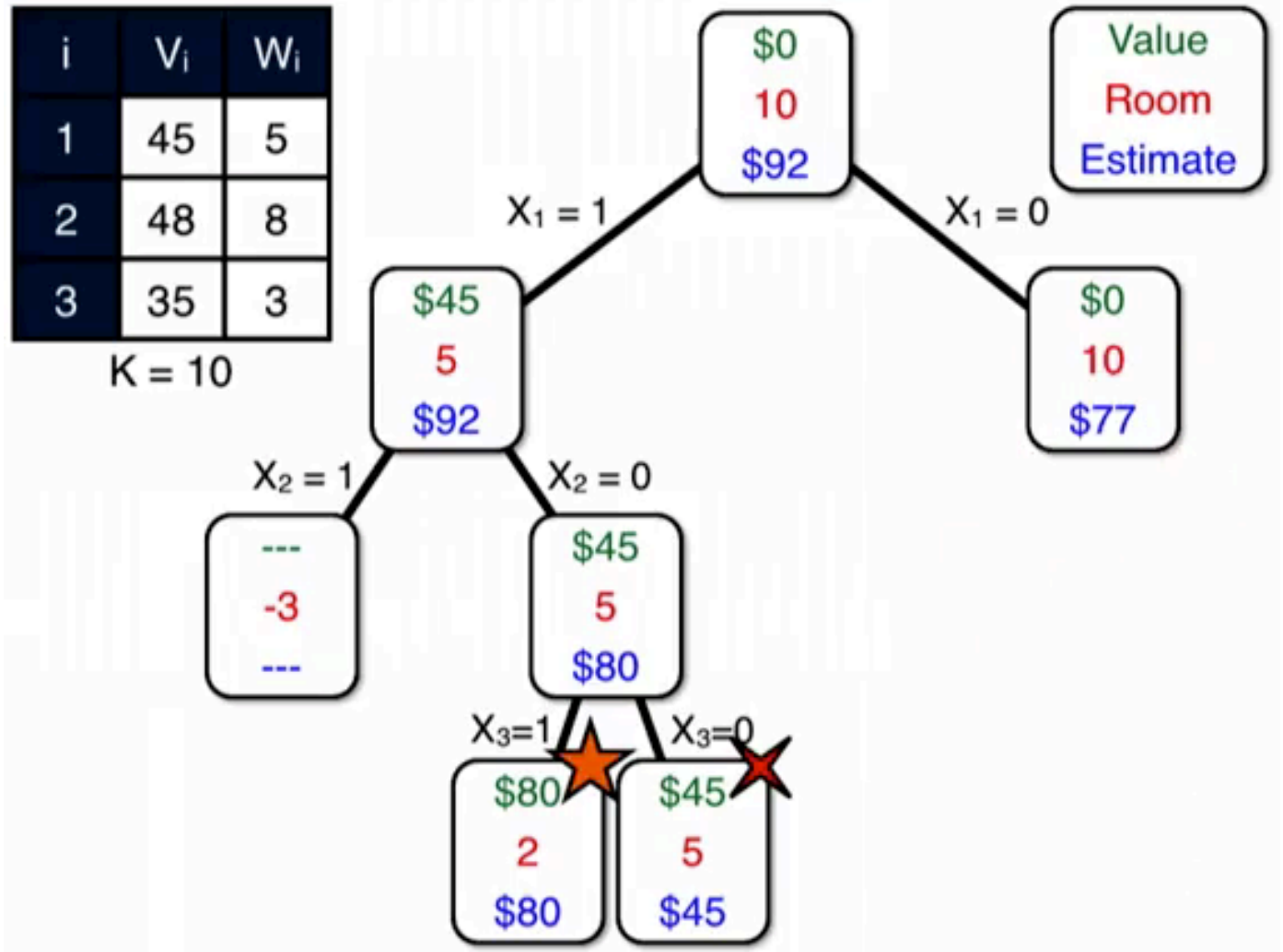
Ramificación y acotación

- Estrategias de búsqueda
  - En profundidad, el mejor primero, menor discrepancia
  - Muchas otras
- En profundidad
  - Poda cuando la estimación del nodo es peor que la mejor solución encontrada
- El mejor primero
  - Selecciona el nodo con la mejor estimación
- Discrepancia limitada
  - Confía en una heurística ávida

Ramificación y Acotación (en profundidad)

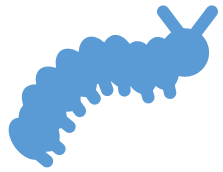
i	V_i	W_i
1	45	5
2	48	8
3	35	3

K = 10



Ramificación y acotación (búsqueda en profundidad)

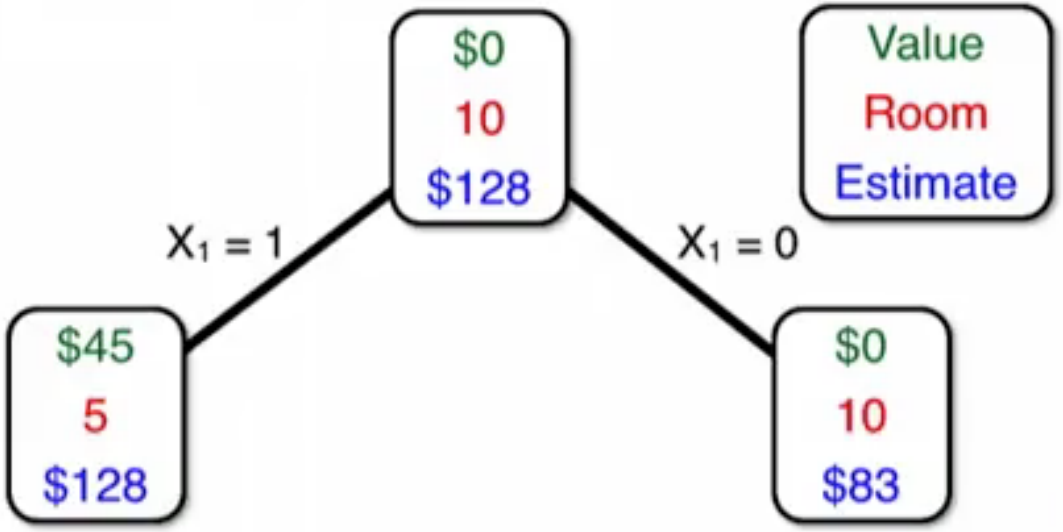
- Ir en profundidad
- ¿Cuándo hace la poda?
  - Cuando encuentra un nuevo nodo peor que la solución encontrada
- ¿Eficiencia en memoria?



Ramificación y Acotación (Mejor primero)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

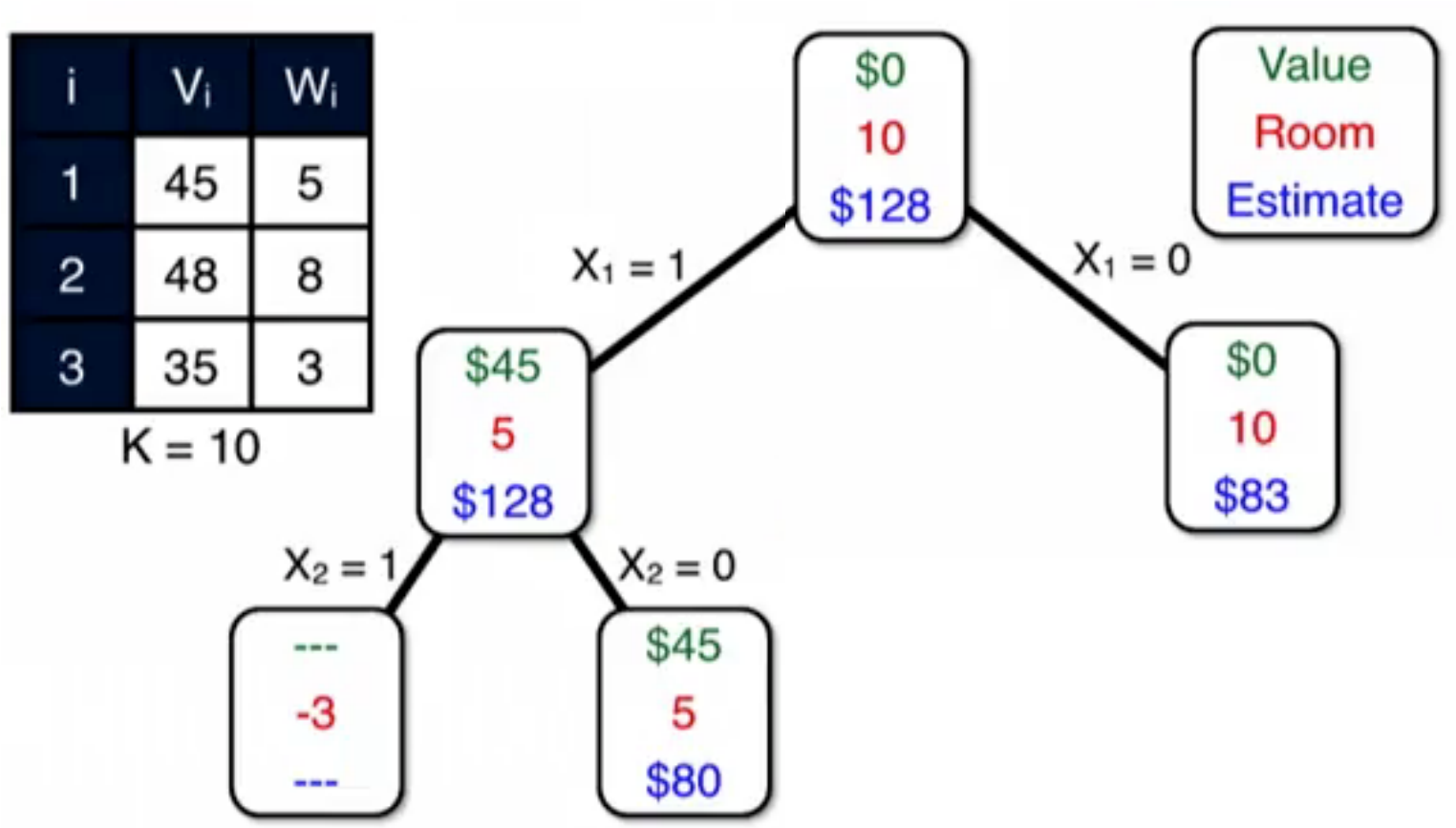
K = 10



Ramificación y Acotación (Mejor primero)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

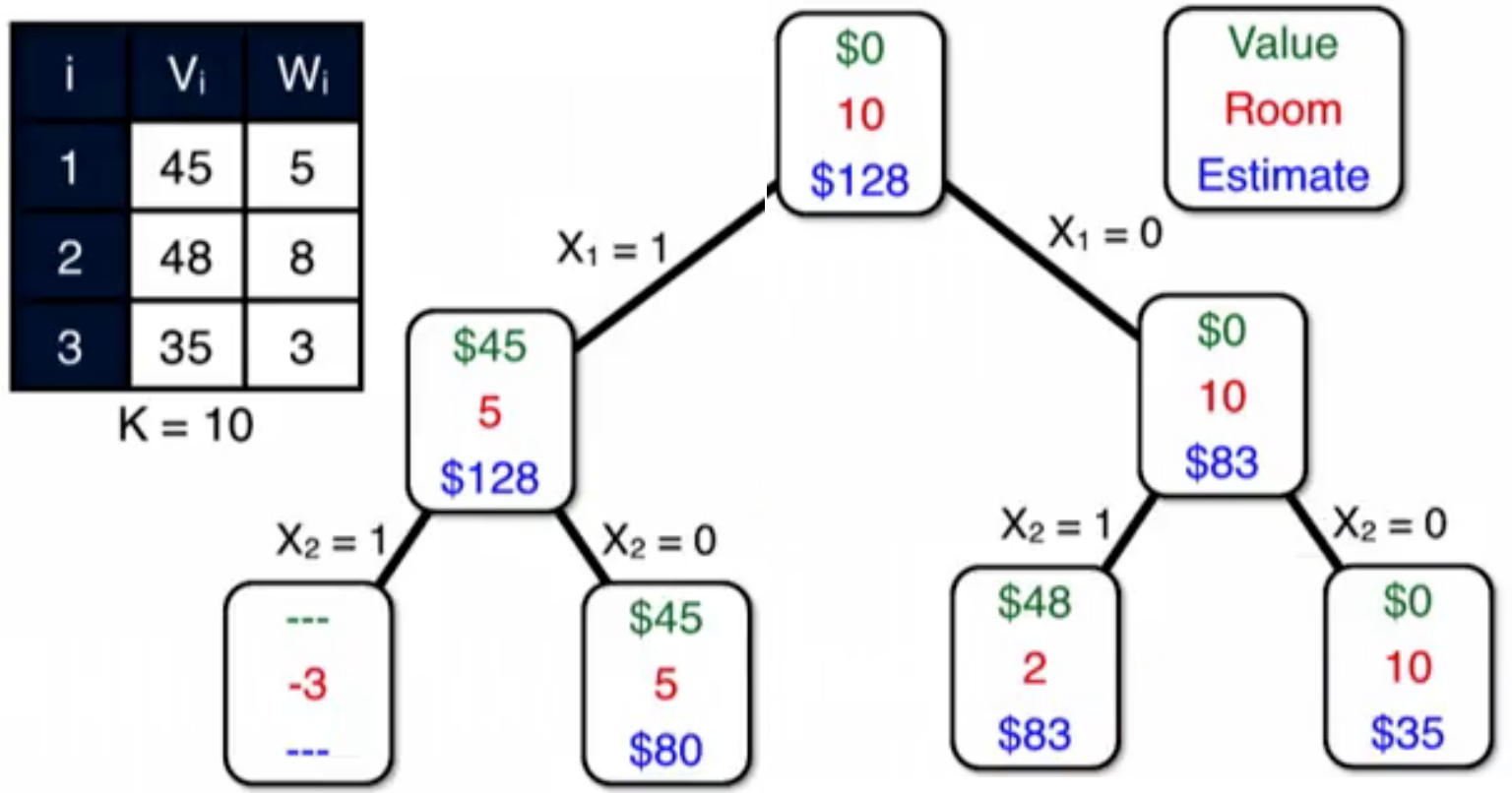
K = 10



Ramificación y Acotación (Mejor primero)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

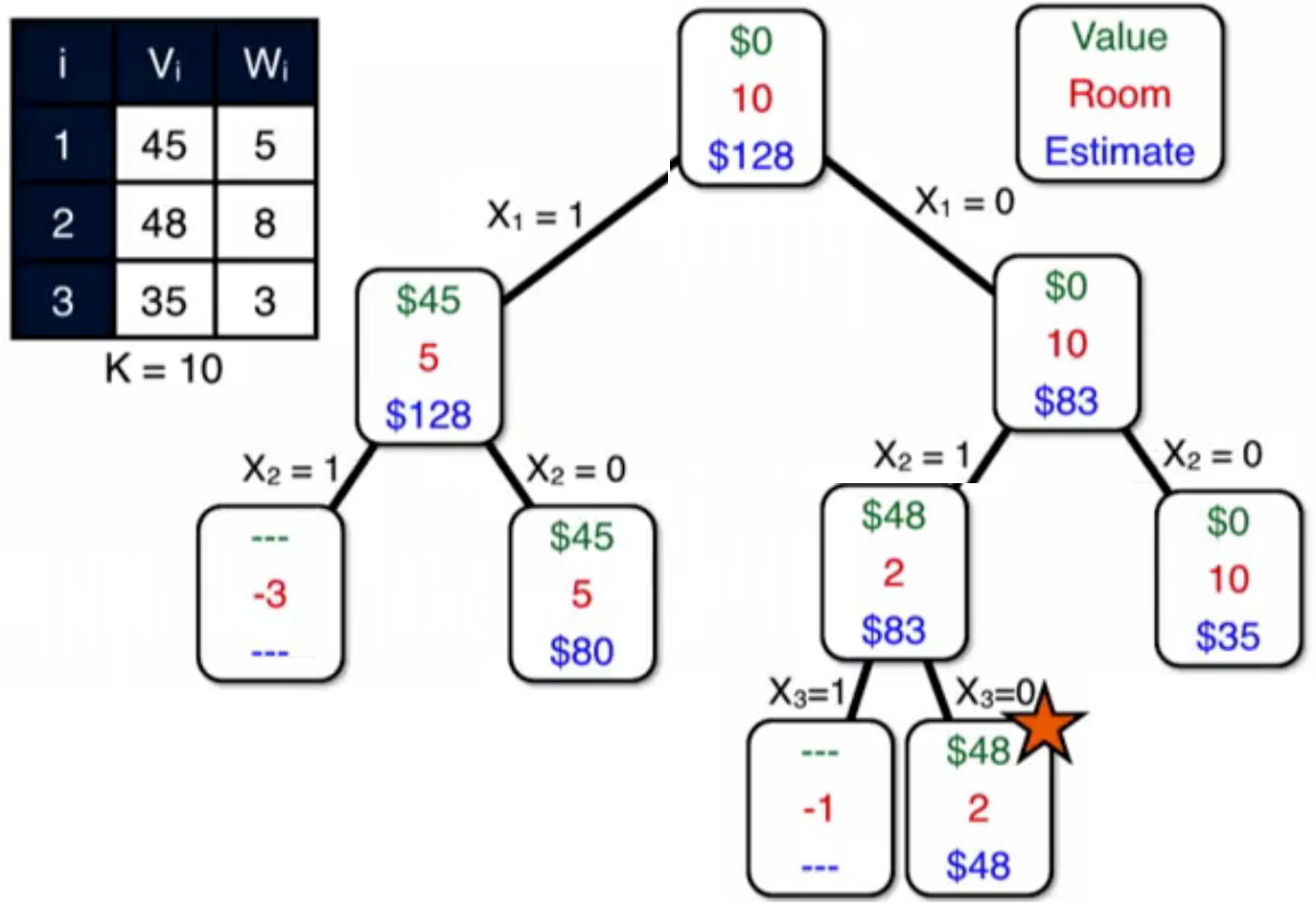
K = 10



Ramificación y Acotación (Mejor primero)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

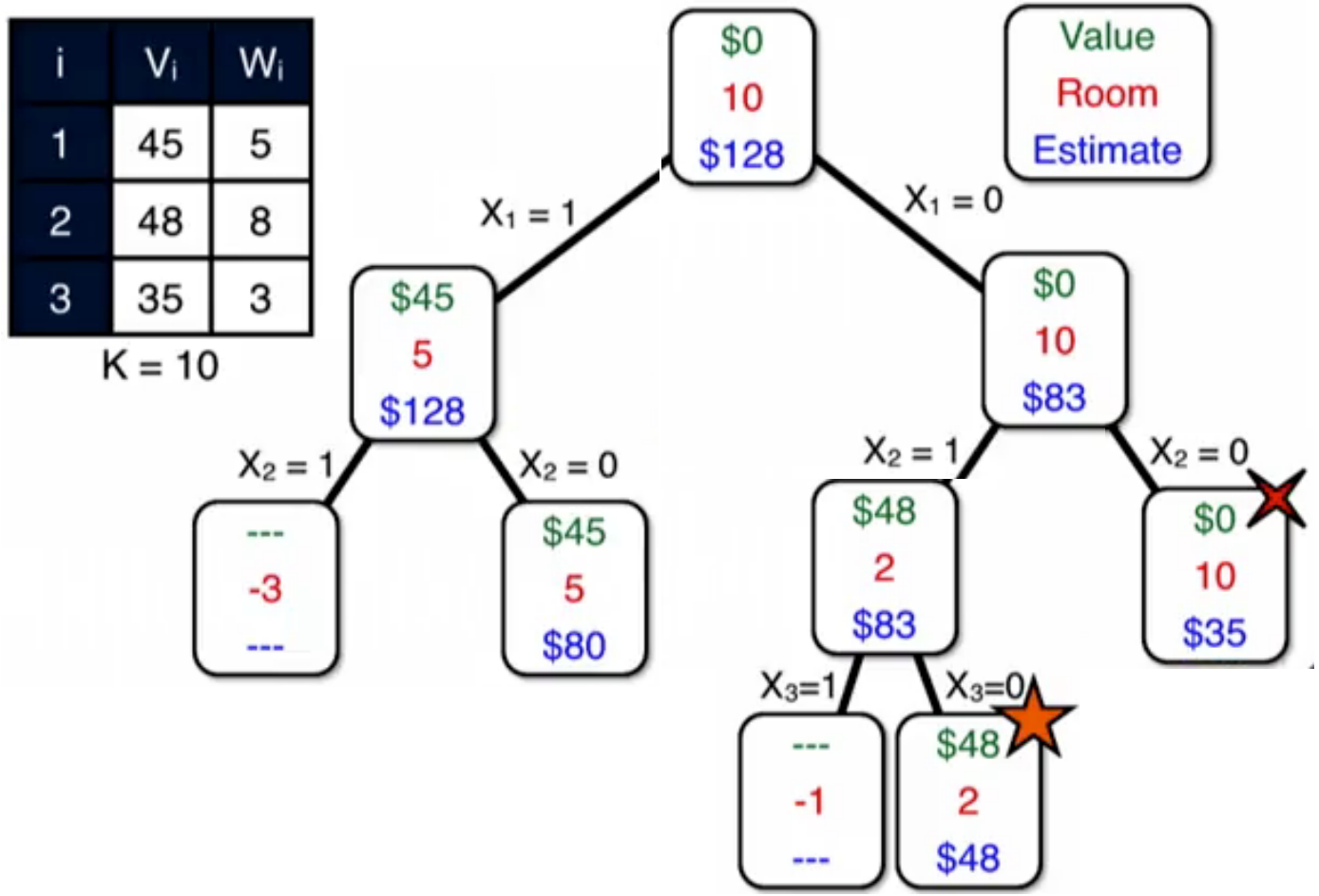
K = 10



Ramificación y Acotación (Mejor primero)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

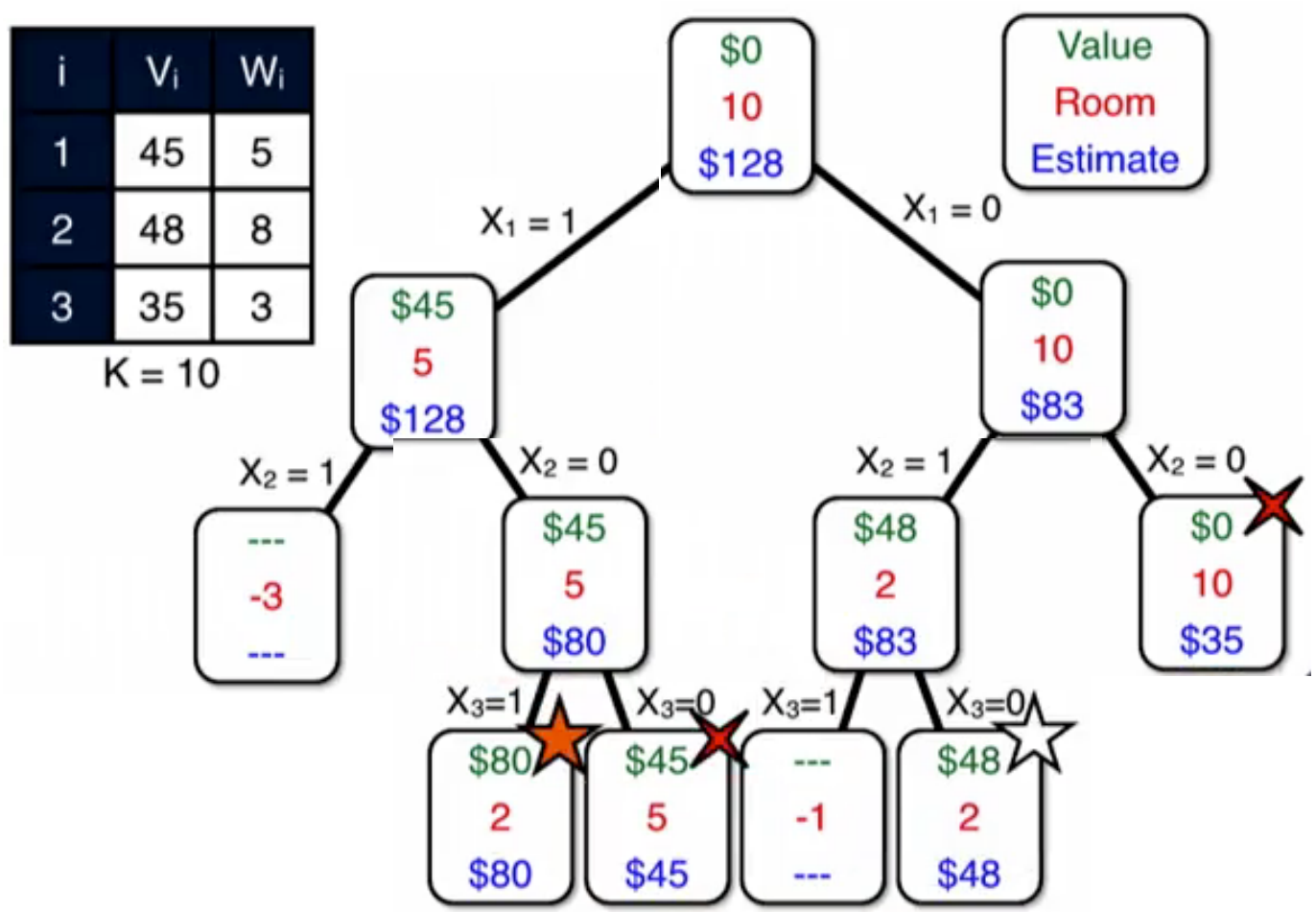
K = 10



Ramificación y Acotación (Mejor primero)

i	V_i	W_i
1	45	5
2	48	8
3	35	3

K = 10

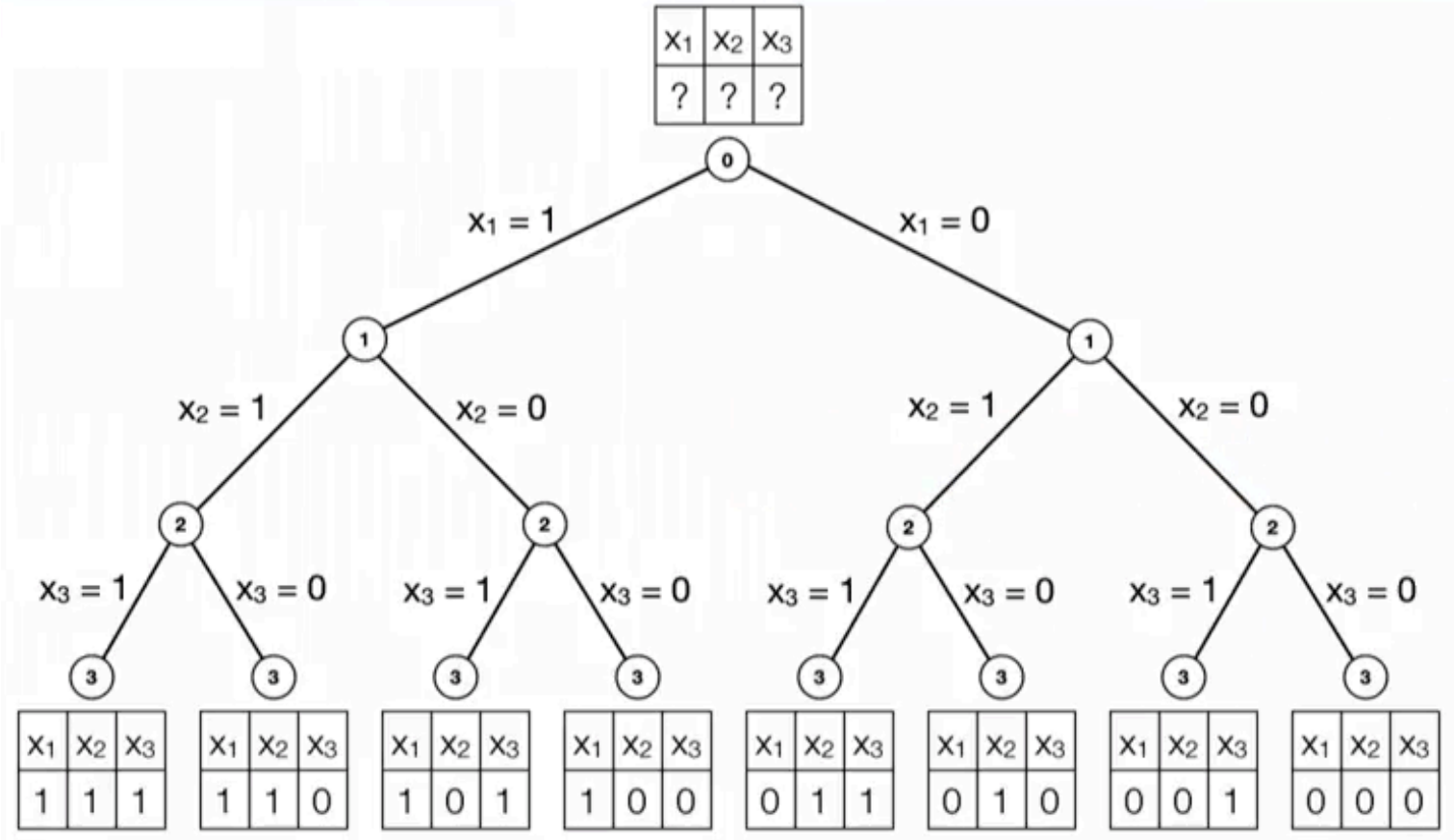


- Ir por el mejor
- ¿Cuándo para?
  - Cuando todos los nodos son peores que la solución encontrada
- ¿Eficiencia en memoria?

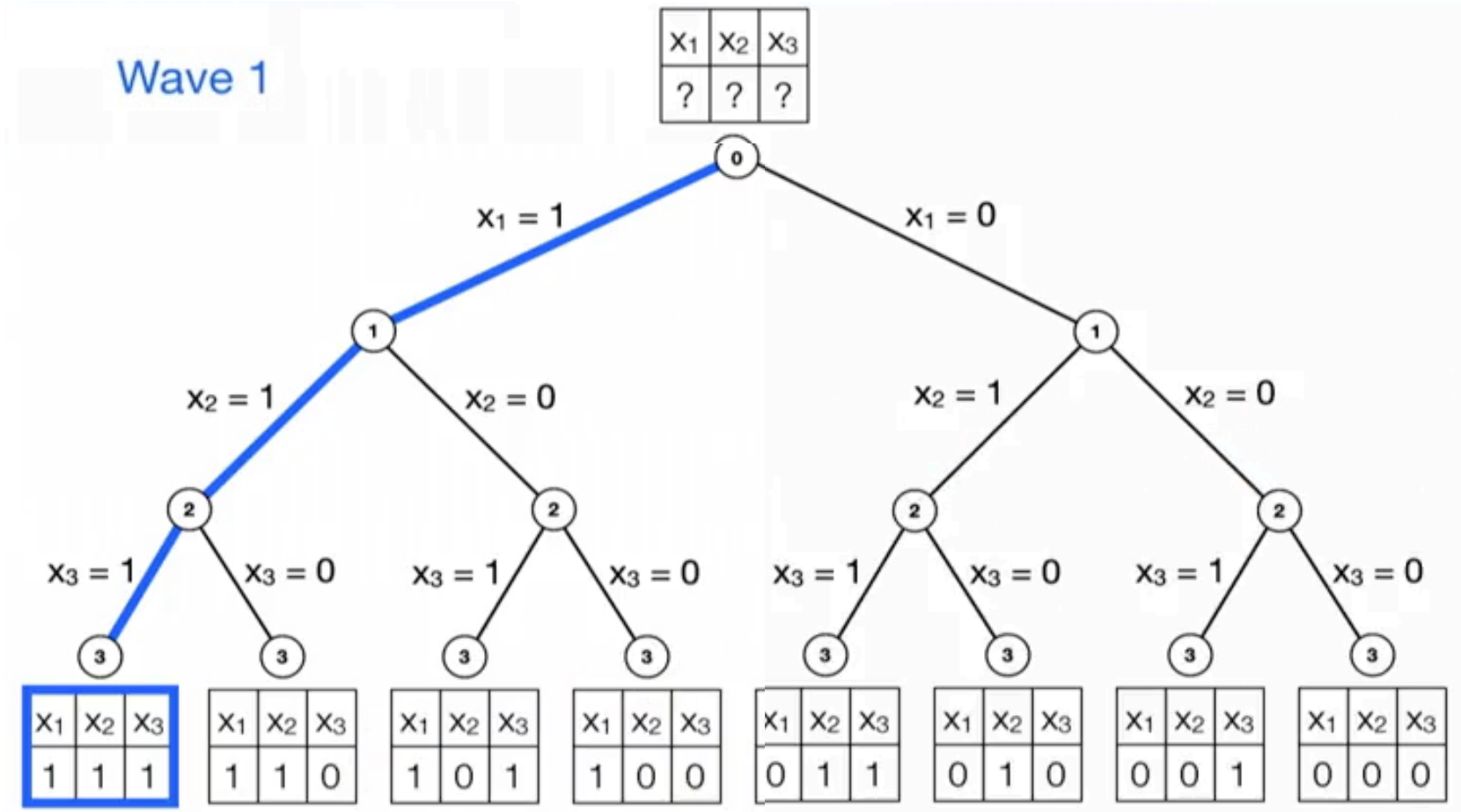
Búsqueda por discrepancia limitada

- Suponer que tenemos una heurística disponible
  - Se cometen muy pocos errores
  - El árbol de búsqueda es binario
  - Seguir la heurística significa ramificar a la izquierda
  - Ramificar a la derecha significa que la heurística era equivocada
- *Limited Discrepancy Search* (LDS)
  - Evita errores
  - Explora el espacio de búsqueda incrementando el orden de los errores
  - Confía cada vez menos en la heurística
- Explora el espacio de búsqueda en olas
  - Sin errores
  - 1 error
  - 2 errores
  - …

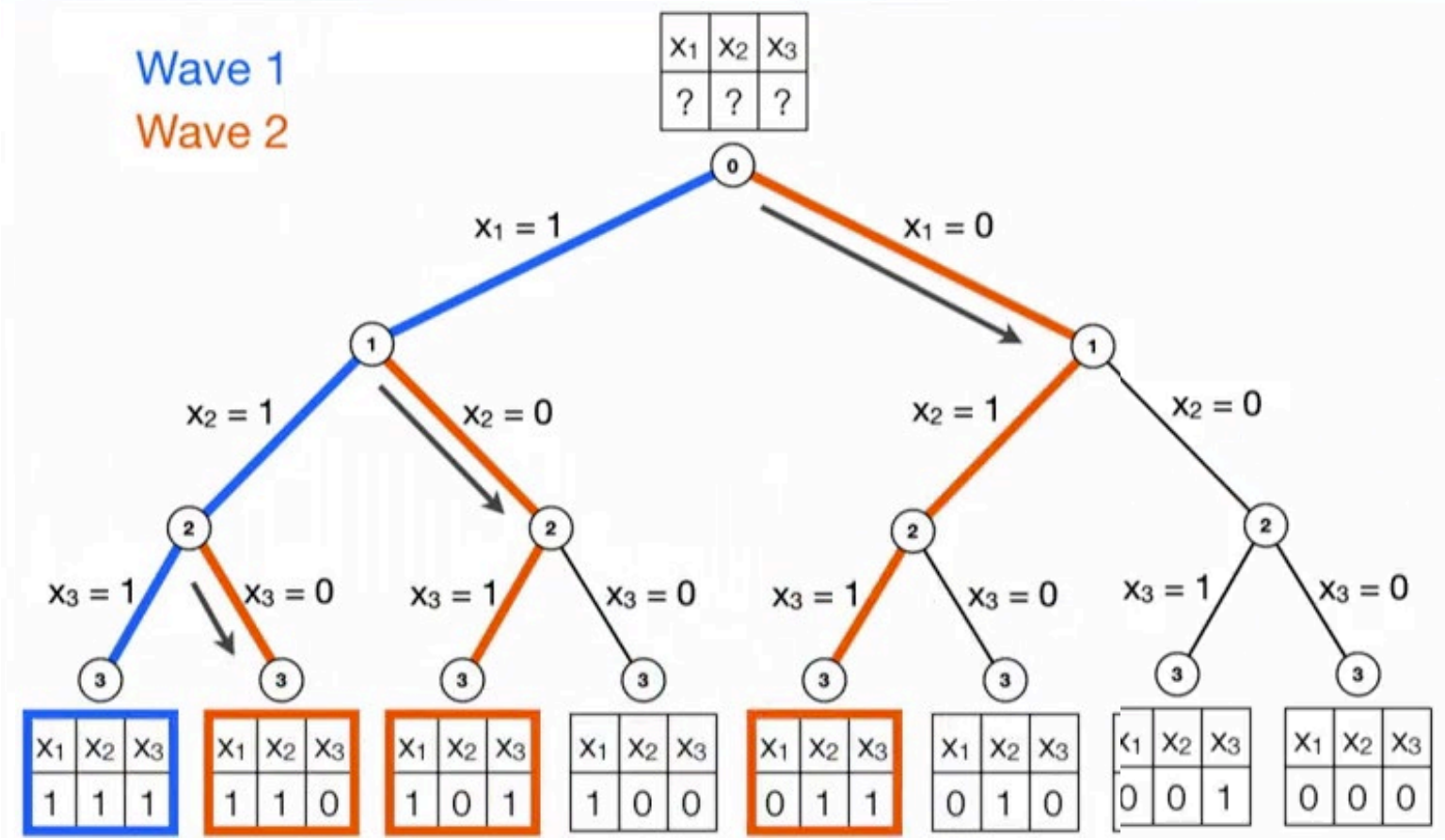
Olas LDS


x1 x2 x3
? ? ?
0
x1 = 1
x1 = 0
1
1
x2 = 1
x2 = 0
x2 = 1
x2 = 0
2
2
2
2
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
3
3
3
3
3
3
3
3
x1 x2 x3
1 1 1
x1 x2 x3
1 1 0
x1 x2 x3
1 0 1
x1 x2 x3
1 0 0
x1 x2 x3
0 1 1
x1 x2 x3
0 1 0
x1 x2 x3
0 0 1
x1 x2 x3
0 0 0

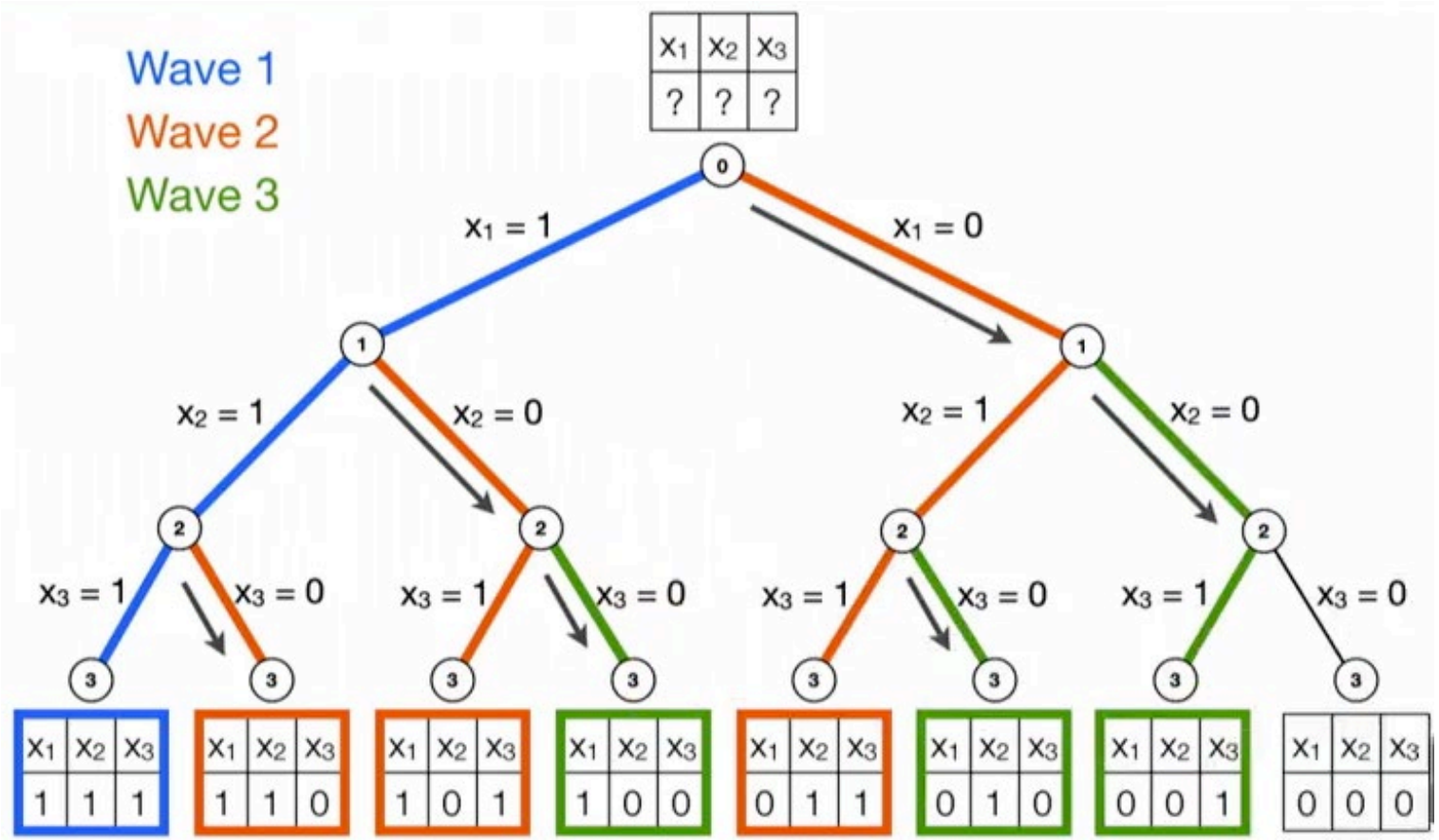
Olas LDS


Wave 1
x1 x2 x3
? ? ?
0
x1 = 1
x1 = 0
1
1
x2 = 1
x2 = 0
x2 = 1
x2 = 0
2
2
2
2
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
3
3
3
3
3
3
3
3
x1 x2 x3
1 1 1
x1 x2 x3
1 1 0
x1 x2 x3
1 0 1
x1 x2 x3
1 0 0
x1 x2 x3
0 1 1
x1 x2 x3
0 1 0
x1 x2 x3
0 0 1
x1 x2 x3
0 0 0

Olas LDS


Wave 1
Wave 2
x1 x2 x3
? ? ?
x1 = 1
x1 = 0
x2 = 1
x2 = 0
x3 = 1
x3 = 0
1 1 1
1 1 0
1 0 1
1 0 0
0 1 1
0 1 0
0 0 1
0 0 0

Olas LDS


Wave 1
Wave 2
Wave 3
x1 x2 x3
? ? ?
x1 = 1
x1 = 0
x2 = 1
x2 = 0
x2 = 1
x2 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
1 1 1
1 1 0
1 0 1
1 0 0
0 1 1
0 1 0
0 0 1
0 0 0

Ramificación y acotación LDS

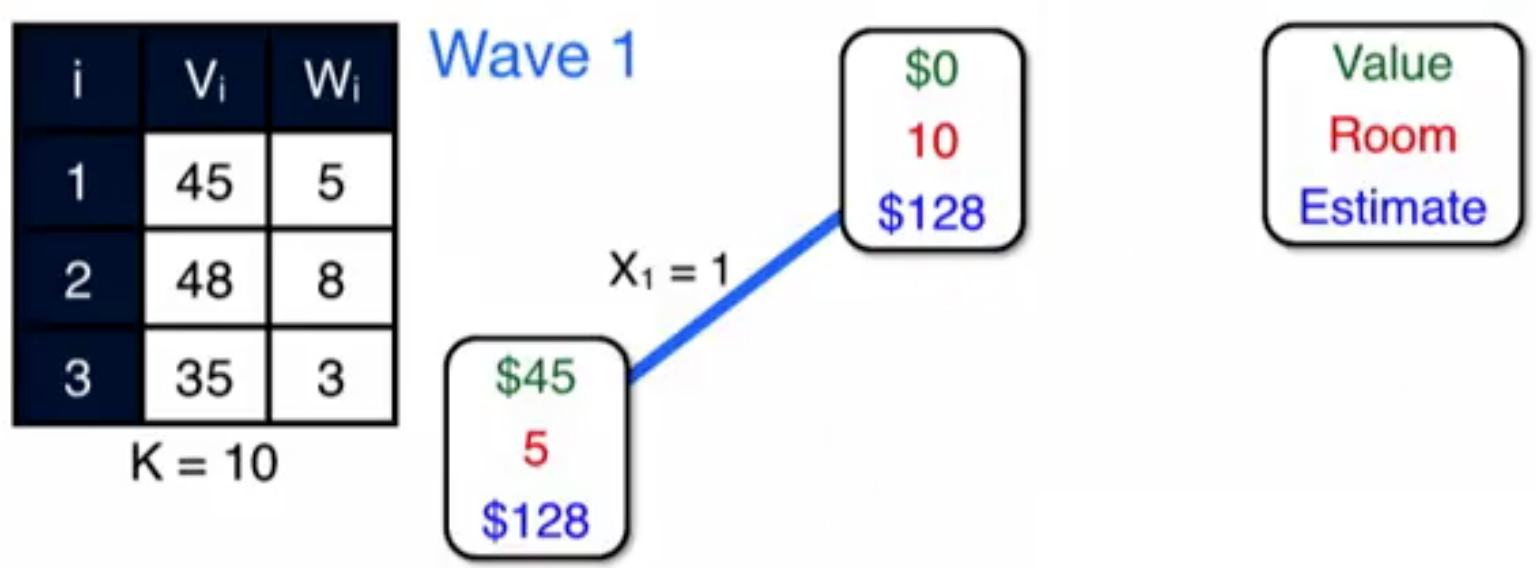
i	V_i	W_i
1	45	5
2	48	8
3	35	3

K = 10

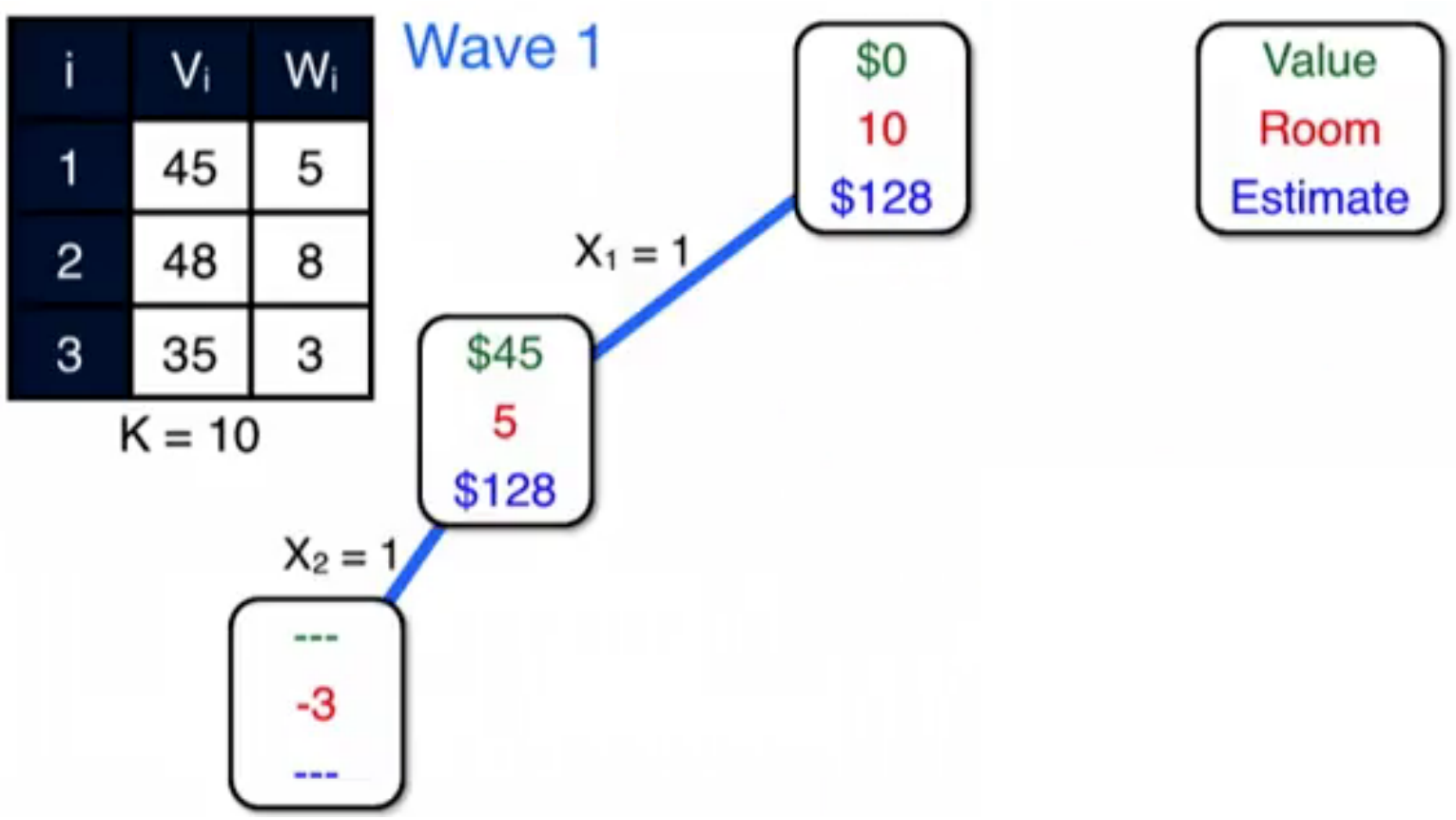
Ramificación y acotación LDS

i	V_i	W_i
1	45	5
2	48	8
3	35	3

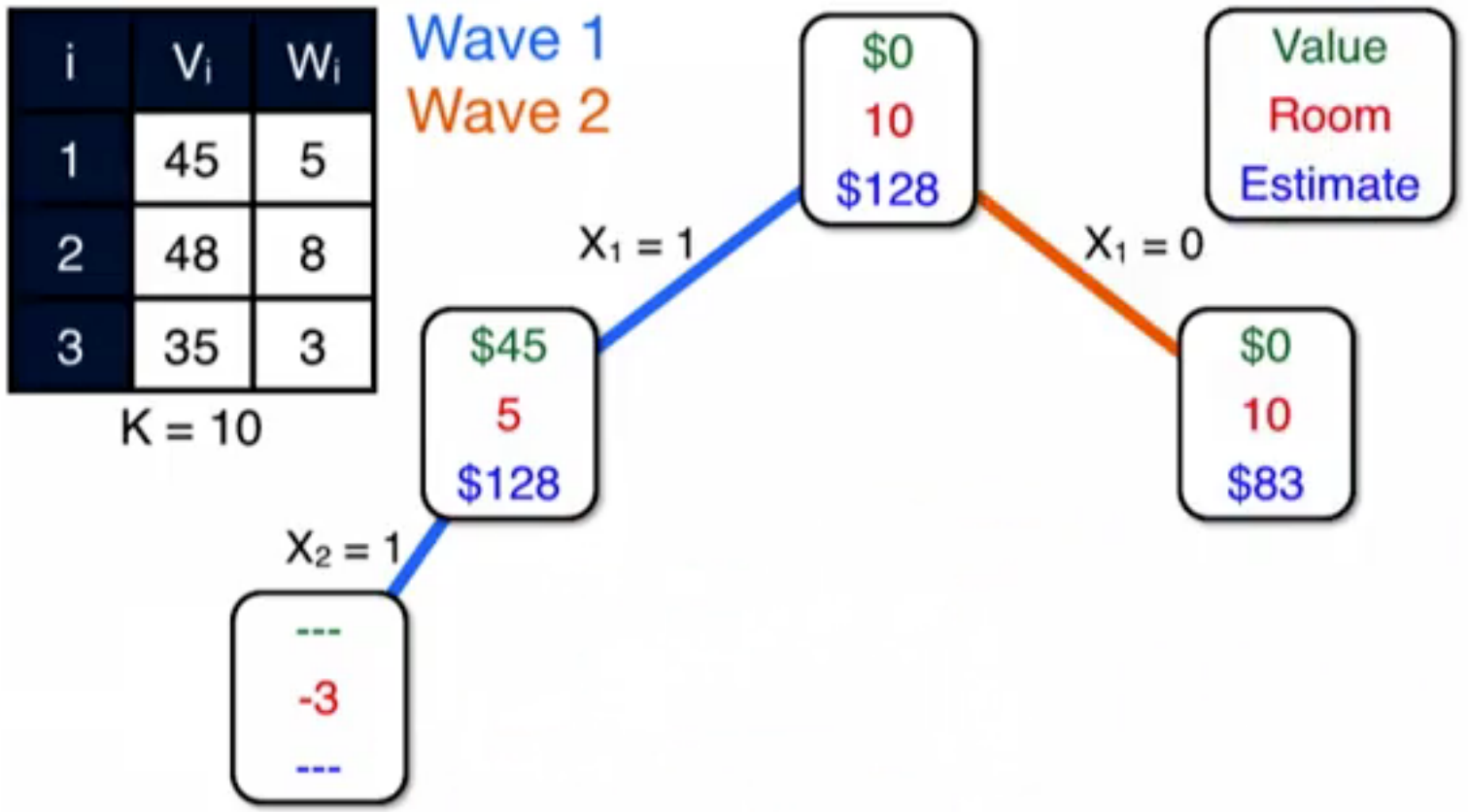
K = 10



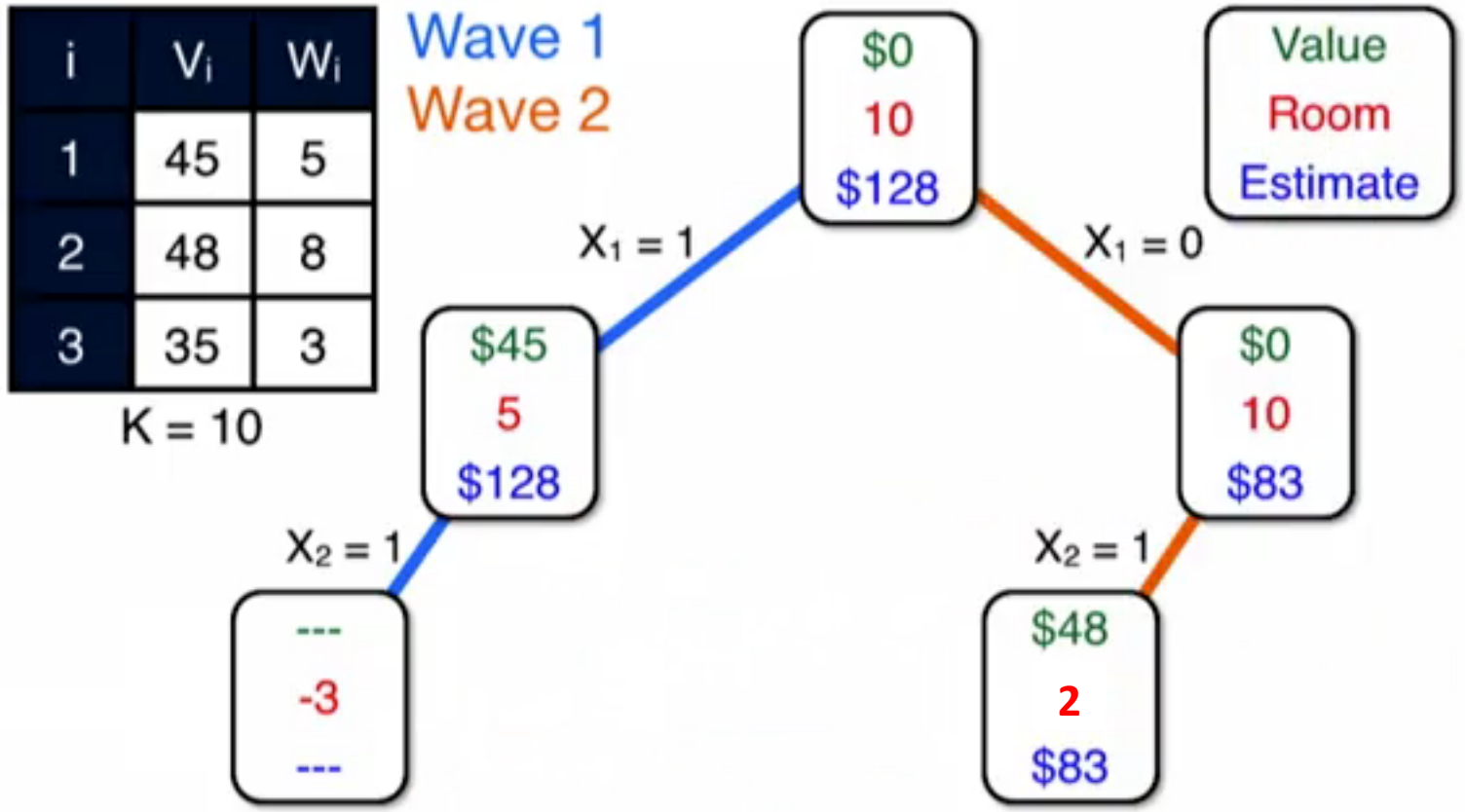
Ramificación y acotación LDS



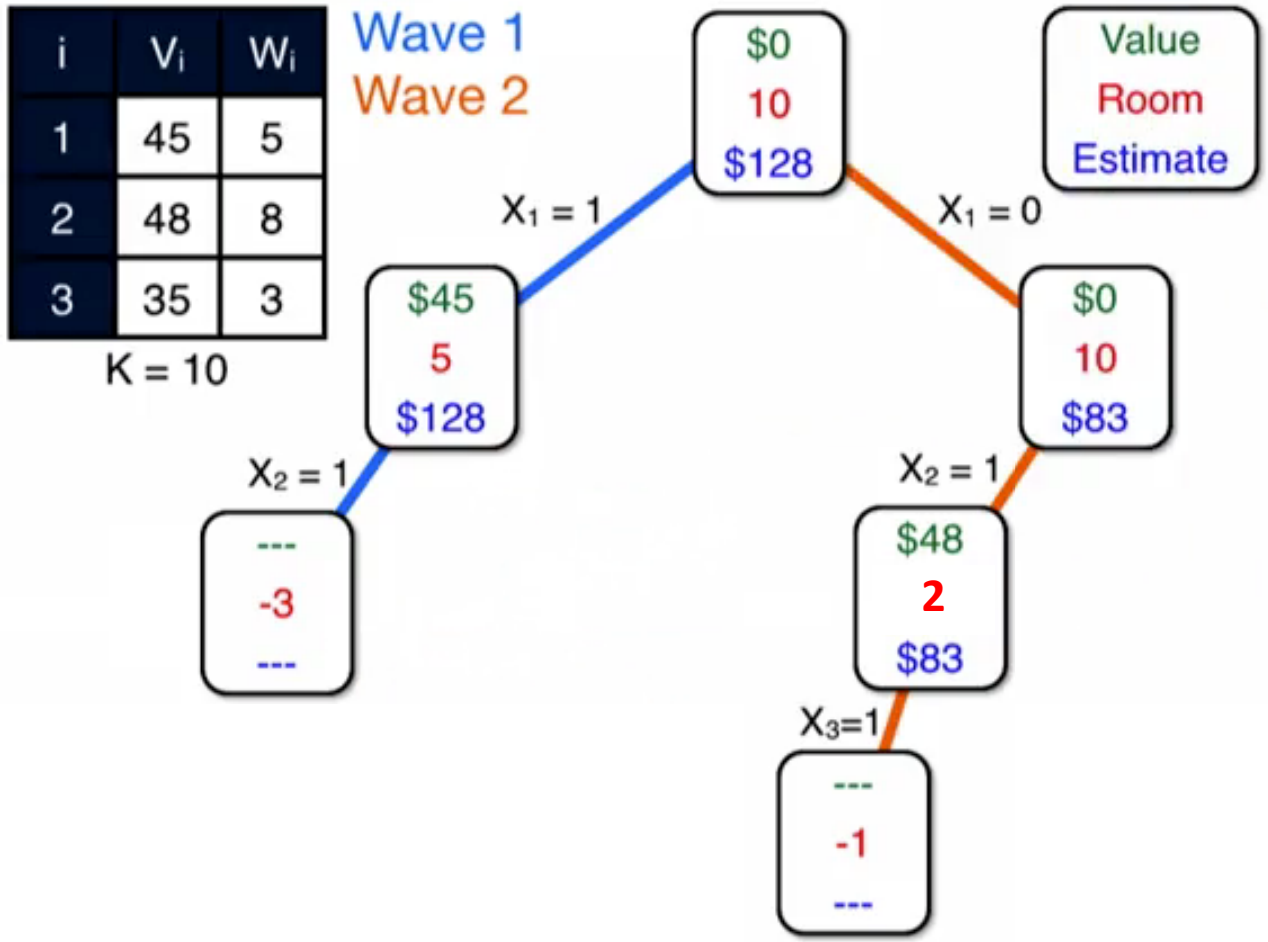
Ramificación y acotación LDS



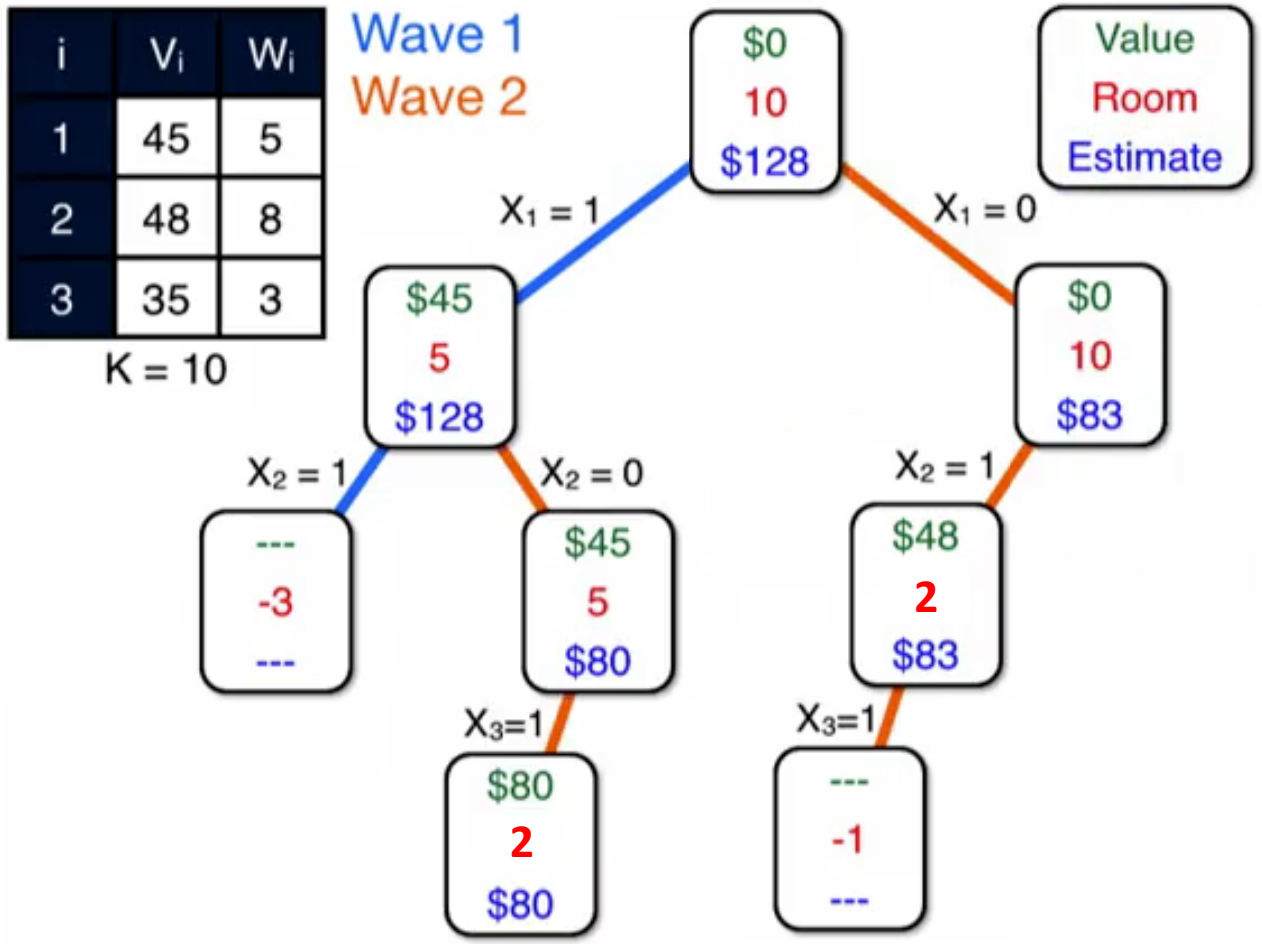
Ramificación y acotación LDS



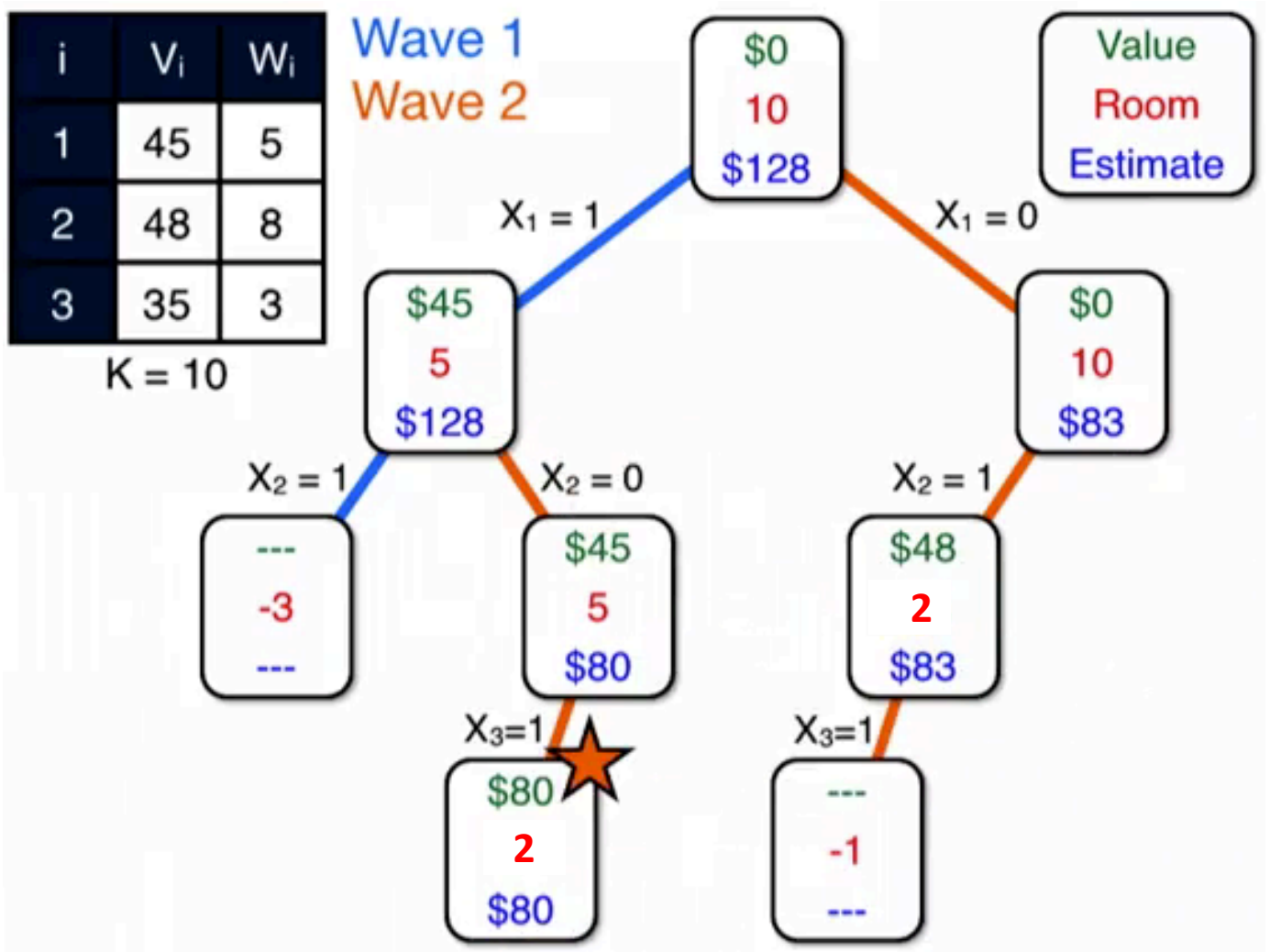
Ramificación y acotación LDS



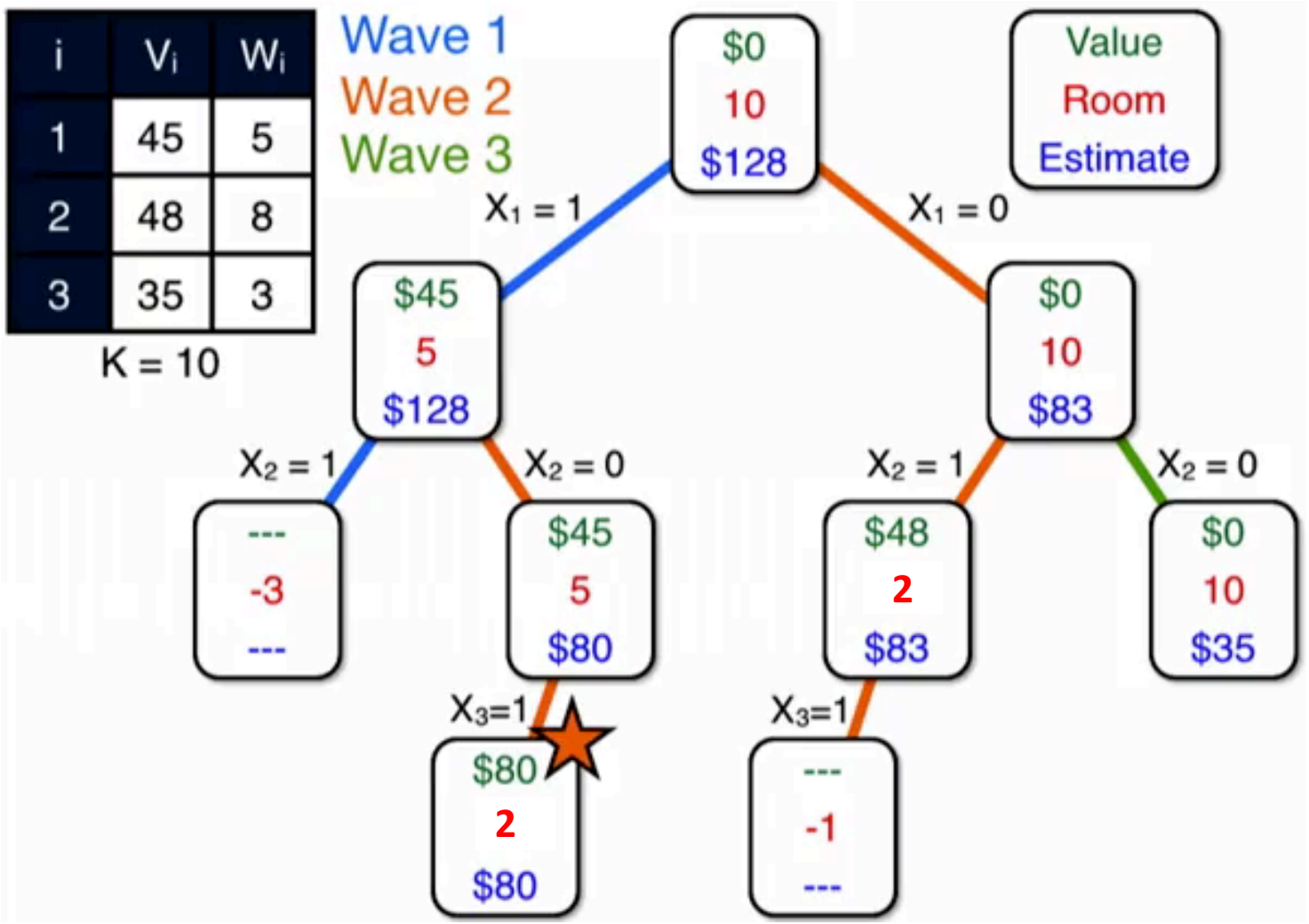
Ramificación y acotación LDS



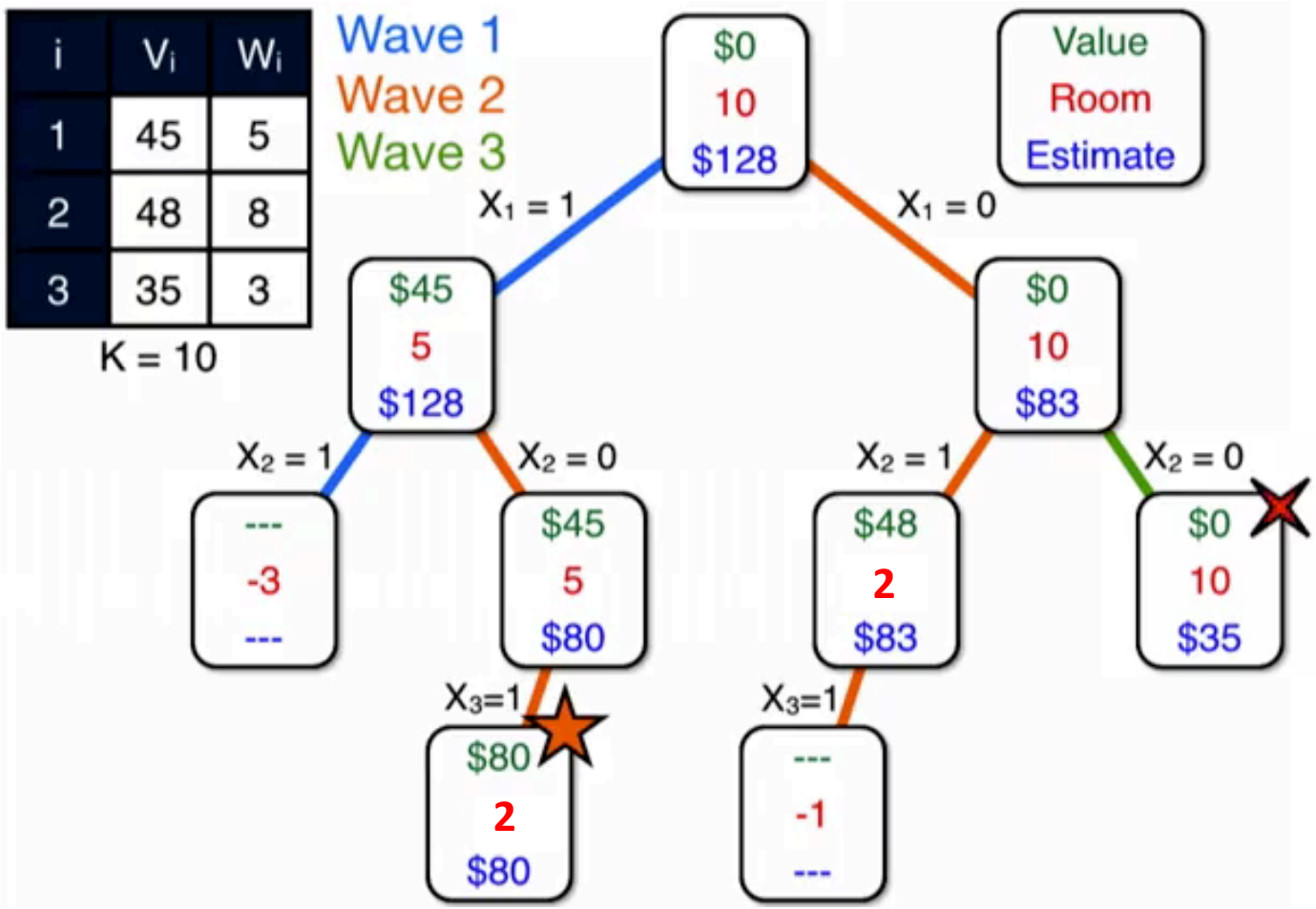
Ramificación y acotación LDS



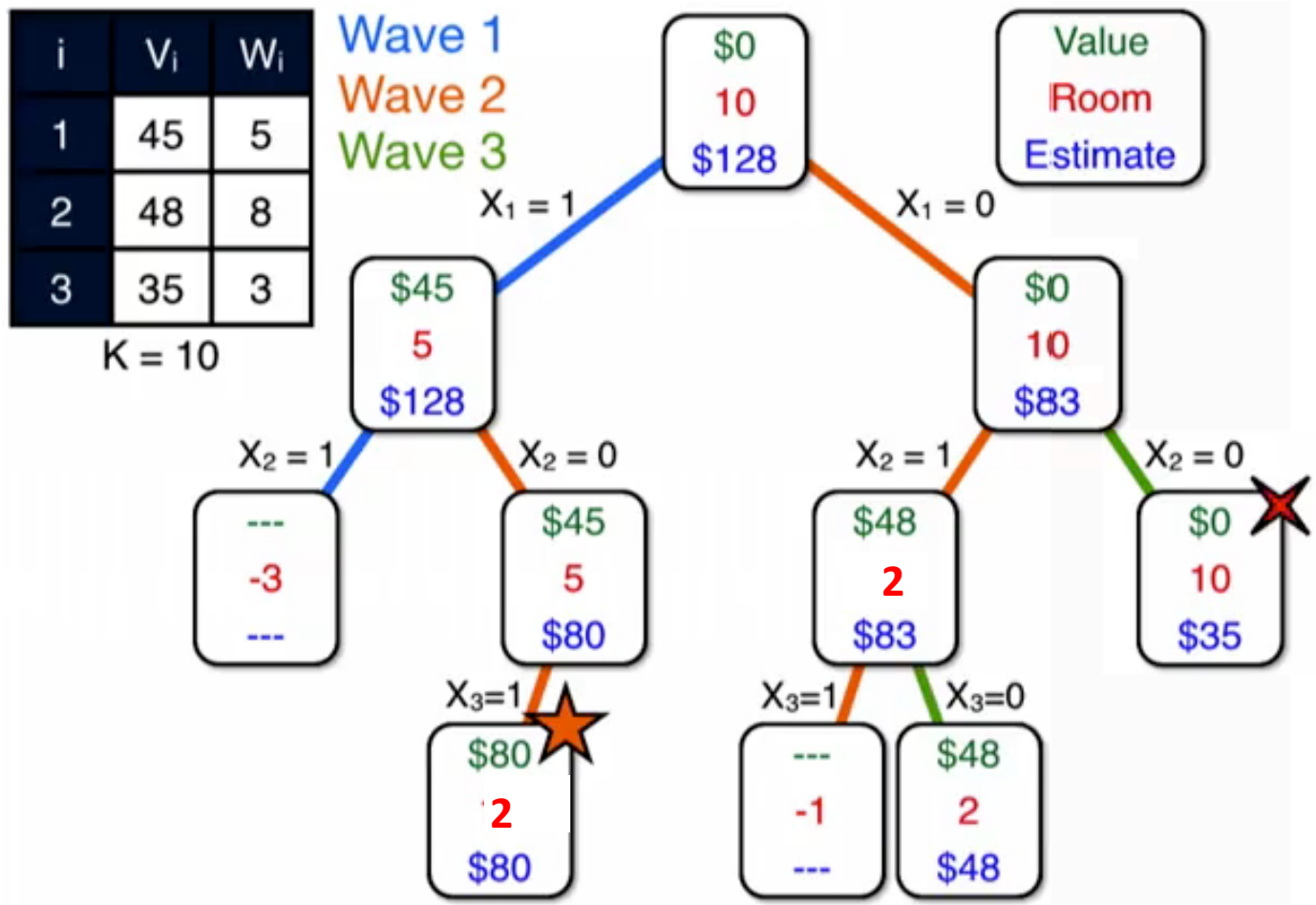
Ramificación y acotación LDS



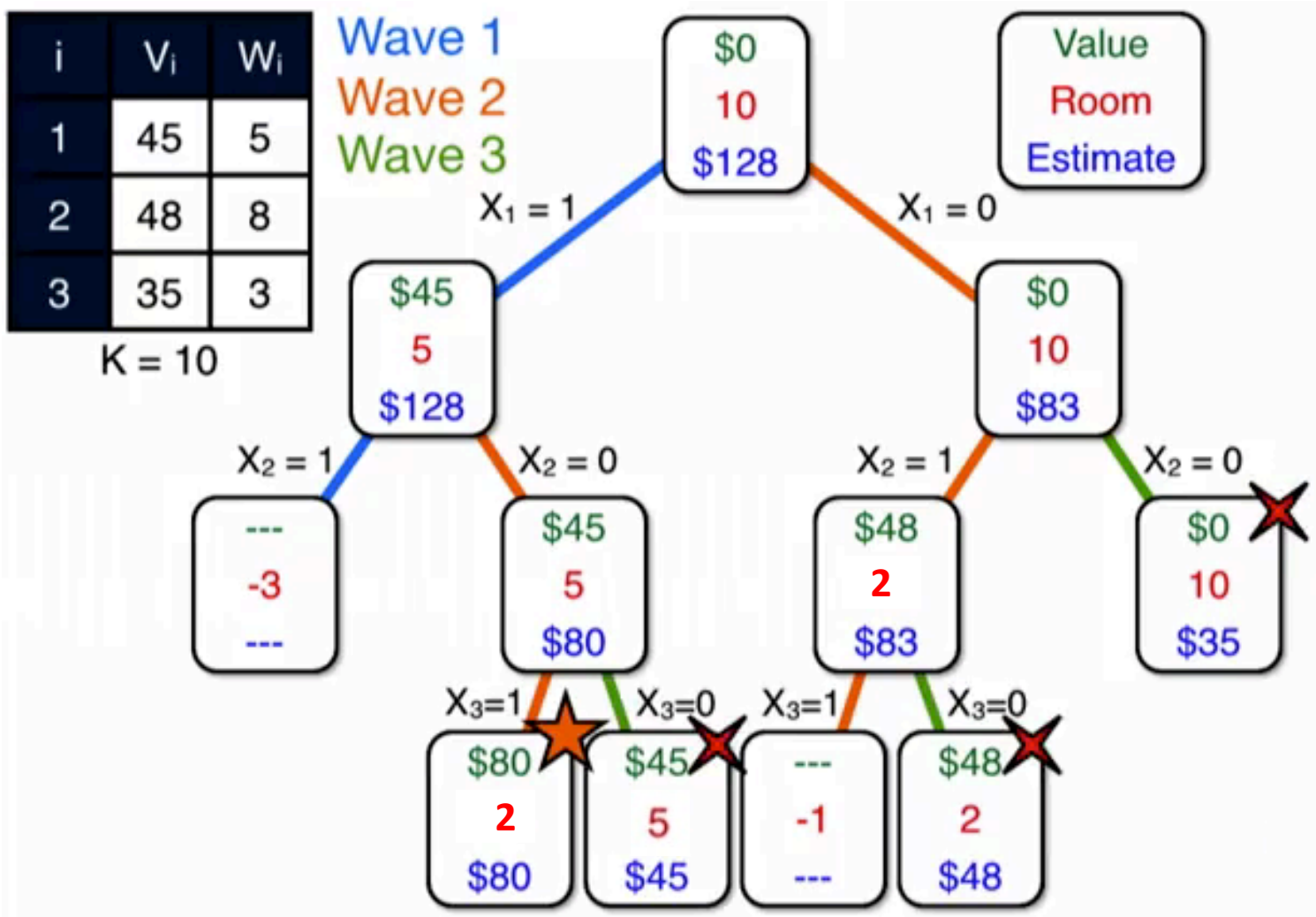
Ramificación y acotación LDS



Ramificación y acotación LDS



Ramificación y acotación LDS

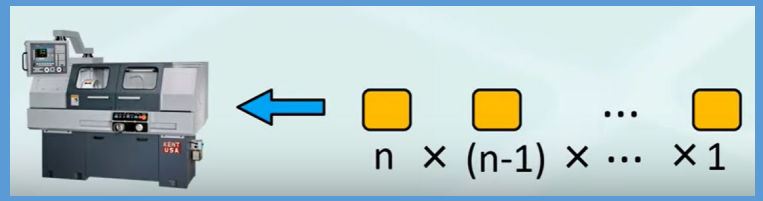


Ramificación y acotación (LDS)

- Confiar en heurística ávida
- ¿Es eficiente en memoria?
  - ¿Comparable a ‘en profundidad’ y primero mejor?
  - Depende de la implementación

Ejemplo de Job Shop *Scheduling (JSP)*

- Asignación de tareas a recursos limitados a lo largo del tiempo
- Ejemplos:
  - Máquinas en un taller
  - Pistas en aeropuertos
  - Unidades de procesamiento en un programa computacional
- Cada tarea puede tener distintos niveles de prioridad
- Una tarea se puede subdividir en subtareas que deben ser ejecutadas de forma secuencial
- Objetivo:
  - Minimizar tiempo de finalización de todas las tareas (*time span*)
  - Minimizar el retraso total de ejecución
  - …



Ejemplo de JSP

- Se desean ejecutar *n tareas (Jobs)*
- Cada tarea tiene un tiempo de ejecución (*Duration*)
- Cada tarea debe ejecutarse antes de una fecha determinada (*Due Date*), a partir de la cual se entra en retraso
- Se desea minimizar el número de días de retraso

Job	Duration (days)	Due Date
A	6	Day 8
B	4	Day 4
C	5	Day 12