Knapsack Problem

Yangqi Su, Yaying Shi

Presentation of the problem

- Input:

  For any n object set, we have size set S, weight set W and value set V for those objects.

  S = $\{s_i | \ where\ i\ from\ 1\ to\ n\} = \{s_1, s_2, s_3, \dots, s_n\}$

  W = $\{w_i | \ where\ i\ from\ 1\ to\ n\} = \{w_1, w_2, w_3, \dots, w_n\}$

  V = $\{v_i | \ where\ i\ from\ 1\ to\ n\} = \{v_1, v_2, v_3, \dots, v_n\}$

  You have a capacity of C_S which is a size limit, C_w is a weight limit.

- Output:

  A subset of S

  T = $\{s_i | \ where\ i \in T\}$

- Metric

  Max$\sum_{i \in T} v_i$

Important of this problem

- Any arrangement and schedule problem with a threshold
  1. If you have an exam tomorrow, how could you select the most value part in limited time.
  2. How could you schedule your travel plan with limited money?
  3. Data download management
  4. Cache select

ILP formulation

$$Maximize \sum_i x_i * v_i \; where \; 1 \leq i \leq n$$

$$subject \; to \; \sum_i x_i * s_i \leq C_S \; where \; 1 \leq i \leq n$$

$$\sum_i x_i * w_i \leq C_w \; where \; 1 \leq i \leq n$$

$$x_i \in \{0,1\}$$

x_i=1 means object was picked, otherwise object was not picked

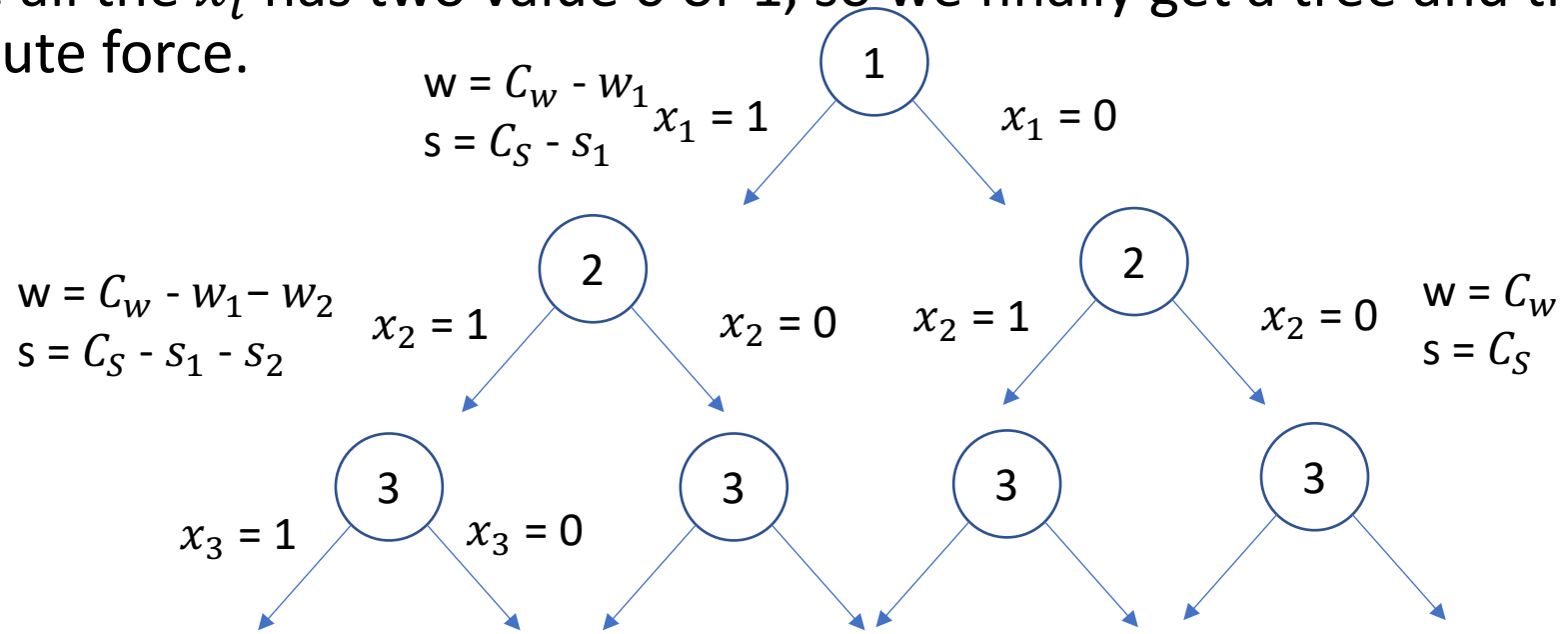
Branch Part: Subset and Solution Space

- Subset

$$S = S_{x_i=0} \cup S_{x_i=1}$$

- Solution Space

  Since all the x_i has two value 0 or 1, so we finally get a tree and the space is 2^n by brute force.



Branching decisions

- We just branching dependent on take the item i or not take the item i.

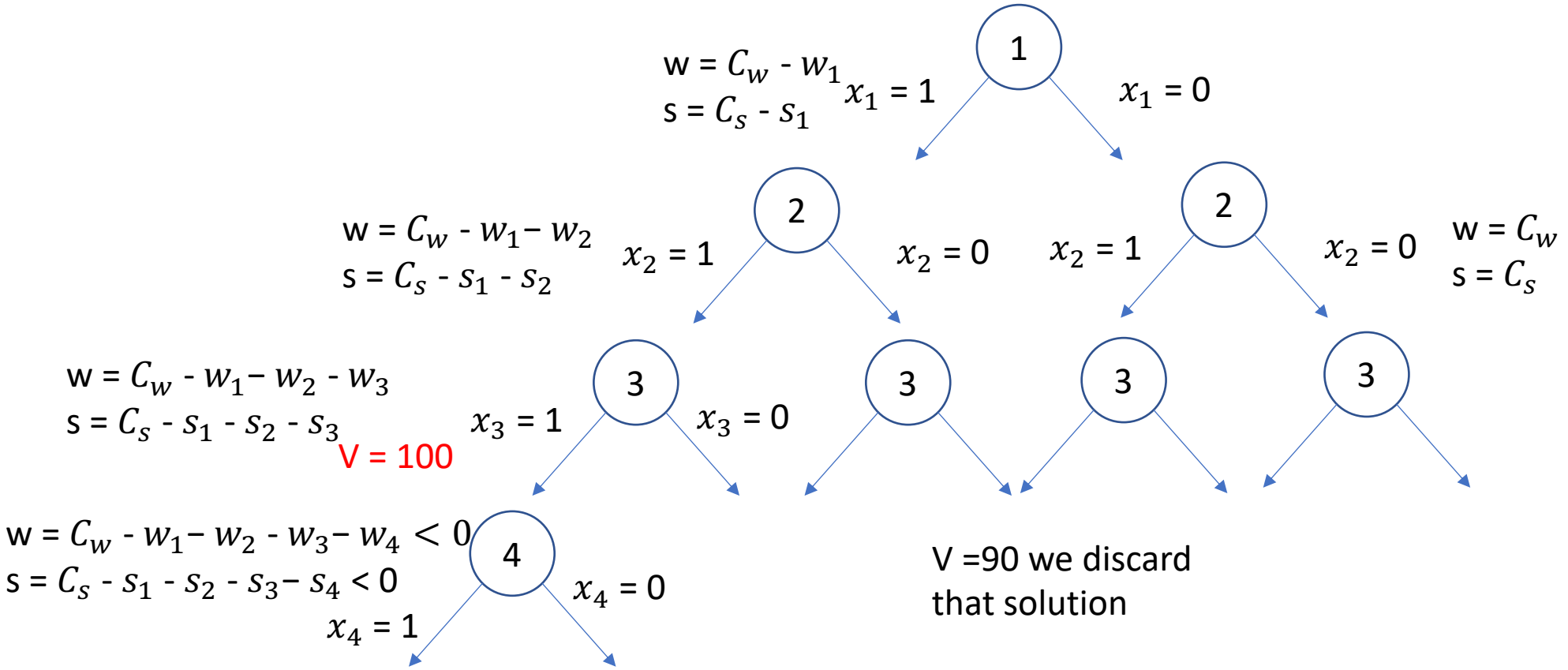
  x_i is a binary variable, we branching according to the value of x_i

- The partition order is determined by the minimal of value/(standardization size) and value/(standardization weight).

  Through the sort, it can do some help to reduce the computation and put the most likely picked item at front.

Bound Part: Bounding

- The bounding gives us one solution that could compared with other solutions.



ILP relaxation

$$Maximize \sum_i x_i * v_i \; where \; 1 \leq i \leq n$$

$$subject \; to \; \sum_i x_i * s_i \leq C_S \; where \; 1 \leq i \leq n$$

$$\sum_i x_i * w_i \leq C_w \; where \; 1 \leq i \leq n$$

$$0 \; \leq x_i \; \leq 1$$

That's means for some item x_i we just take part of its value.

Why the relaxation ILP works

We can change any of the problem to a new form by Let $x_i = \frac{y_i}{v_i}$

$$Maxmize \sum_i x_i * v_i = \sum_i y_i \; where \; 1 \leq i \leq n$$

$$subject \; to \; \sum_i x_i * s_i = \sum_i \frac{s_i}{v_i} * y_i \leq C_S \; where \; 1 \leq i \leq n$$

$$\sum_i x_i * w_i = \sum_i \frac{w_i}{v_i} * y_i \leq C_w \; where \; 1 \leq i \leq n$$

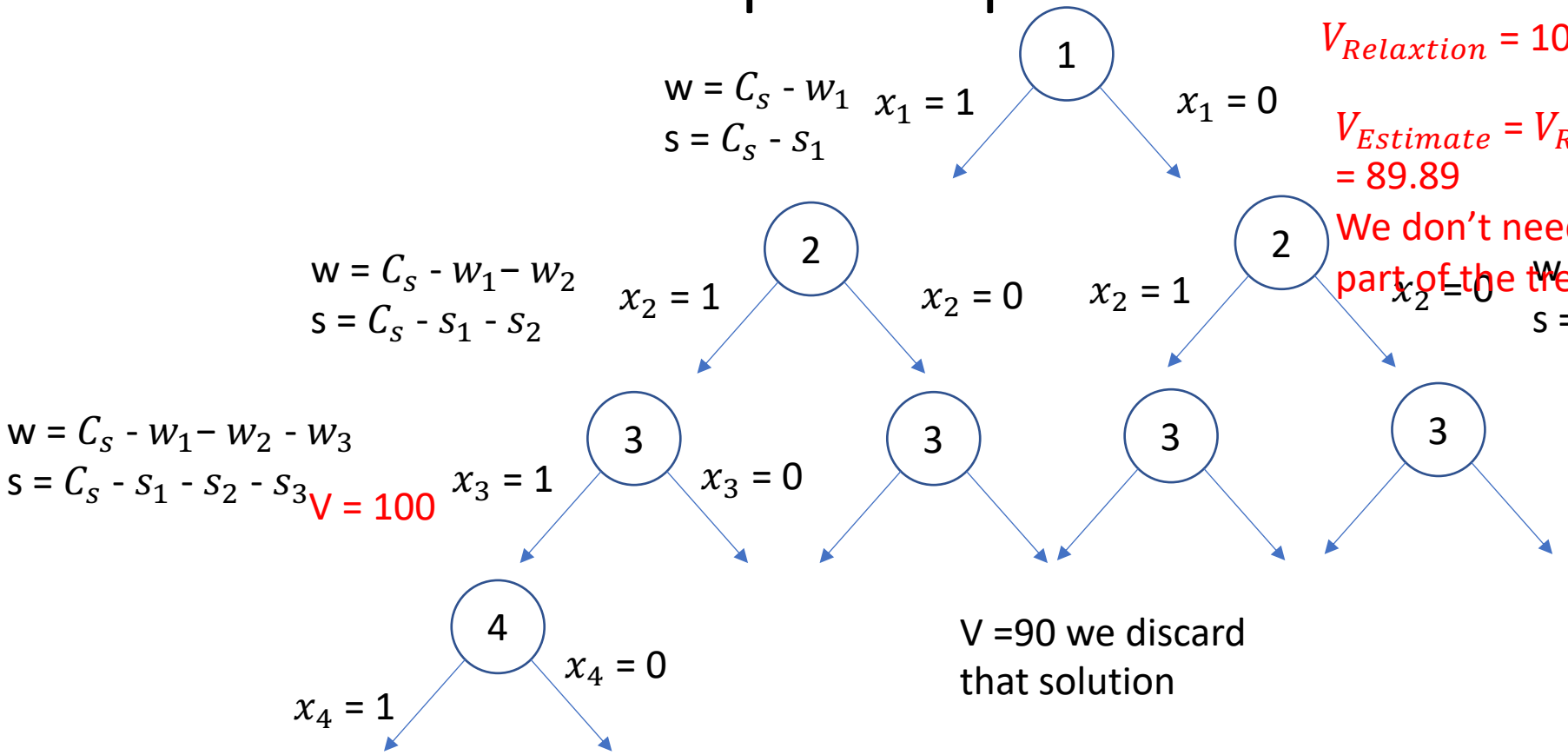
$$0 \; \leq y_i \; \leq \; v_i$$

But we know we can break an item into part, so the relaxation gives the max value we can use for the relaxation problem and a real problem could never reach it.

The relaxation problem

- Pick only one of the constraining dimension and solve exactly.
- pick only one of the constraining dimension and solve fractionally with largest ratio first.
  - Use the greedy algorithm meet the size requirement and the rest of size room use fraction on rest item to get the max value V_{s_max}
  - Use the greedy algorithm meet the weight requirement and the rest of weight room use fraction on rest item to get the max value V_{w_max}
  - Use minimal value of two estimate value min(V_{s_max}, V_{w_max})

How could this help our problem?



$V_{Relaxtion}$ = 109.89 ($x_1 = 1$)

$V_{Estimate}$ = $V_{Relaxtion}$ - V_1 = 109.89 – 20 = 89.89

We don't need to expand the whole right part of the tree.

Trade off Part: Traversal order

Breadth first search

It makes less sense since you don’t know where is the threshold stop. You may expand a lot of the useless node and use a lot memory since you need remember everything in the solution tree. Your memory space will be 2^n.

1) Stop and back

When you reach the threshold, or you find the its solution is bad than current best solutions.

2) Memory Space

2^n

Traversal order

Depth first search

It makes more sense than Breadth since you save some memory space. You only need to remember the node you expended.

1) Stop

When you reach the threshold, or you find the its solution is bad than current best solutions.

2) Memory Space

2 * n

Traversal order

Uniform Cost Search vs Best First Search(A^*)

Always search by the best estimated value. Since we are in the tree, the cost for each steps is 1. So, uniform cost search and best first search is almost same in this solution tree.

1) Stop

When you reach the threshold, or you find the its estimate value is bad than current best solutions.

2) Memory Space

The memory space is large, almost same as Breadth first search. $b^{[C^*/\varepsilon]}$

where C^* is the cost of the optimal solution, every action costs at least ε.

Traversal order

Least Discrepancy Search

First you pick all left branches, then each time you choose one more right branch.

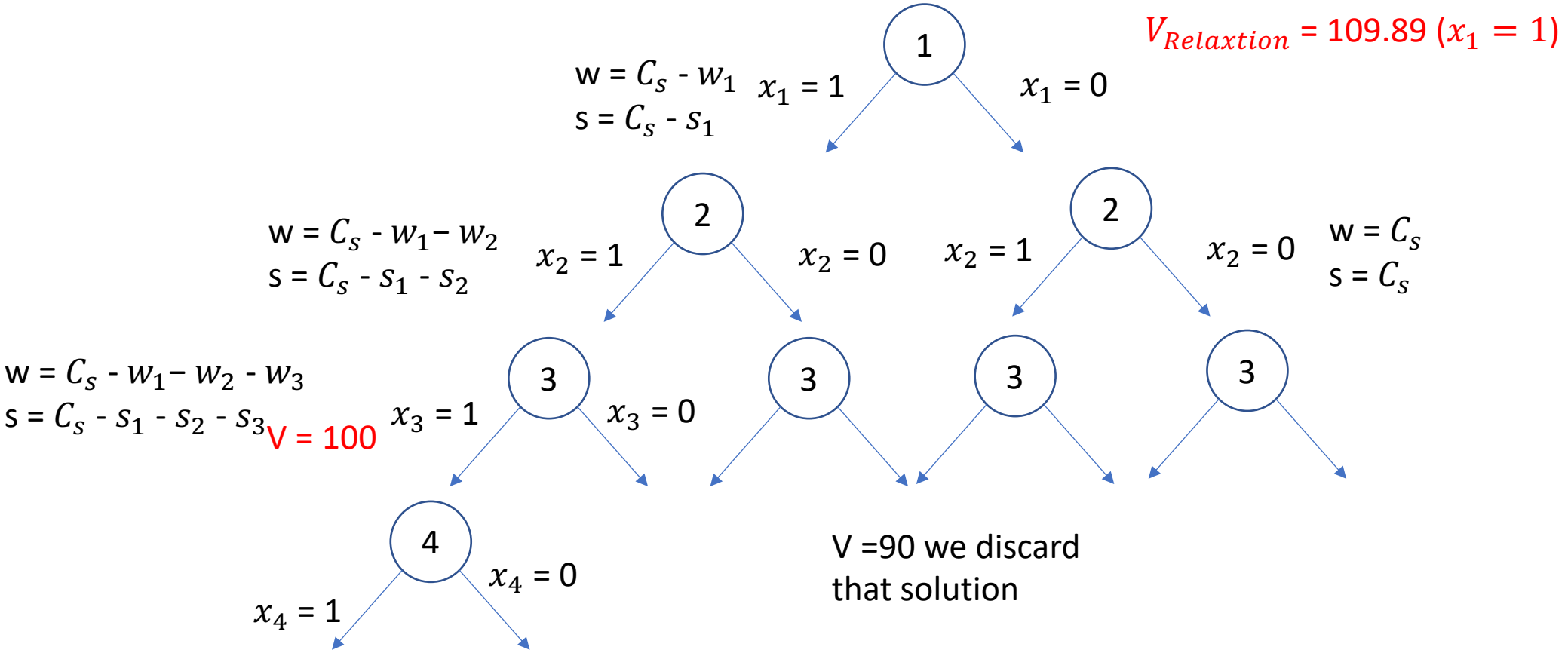
1) Stop

Your estimate value is bad than current best solutions.

2) Memory Space

C_n^x $(x\ is\ times\ you\ want\ go\ right)$

$Do\ a\ V_{Estimation}$



Symmetries or Dominance properties

If an object is heavier and bigger and less valuable than an other one, we won‘t pick it unless we picked the other one.

Deriving a solution from a particular subset

For one subset, we get the solution and record the branch choose, left means picked, right means not pick, we can get a solution by mapping branch to 0,1, and get the solution.